CURSO PRÁTICO 39 DE PROGRAMAÇÃO DE COMPUTADORES

INPUT

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

NOVA CULTURAL

Cz$ 20,00

INPUT

Vol. 3 Nº 39

NESTE NÚMERO

CÓDIGO DE MÁQUINA

AS INSTRUÇÕES DO JOGO

Impressas na tela, as informações contidas no programa deste artigo permitirão que ajudemos o indigitado Willie em sua incansável busca do lanche roubado. Aprenda ainda a chamar sub-rotinas da memória ROM 761

PROGRAMAÇÃO BASIC

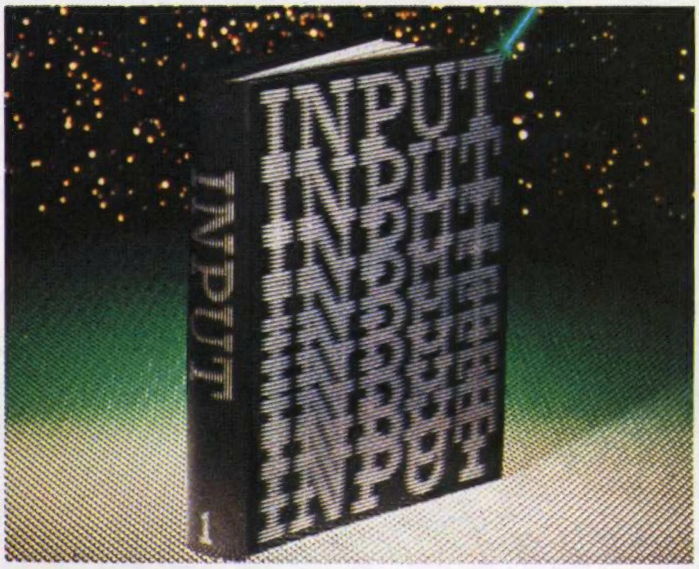
TUDO QUE SOBE, DESCE...

Como desenhar a trajetória de projéteis. Efeitos de diferentes campos gravitacionais. Aprenda a simular parábolas. Movimento vertical e movimento horizontal. Como unir as rotinas. Mude o ângulo 766

PROGRAMAÇÃO BASIC

ACASO E PROBABILIDADE

O que é probabilidade e como medi-la. Probabilidade, freqüência e acaso. Um programa lançador de moedas. Probabilidade de vários resultados. Curva de distribuição. O triângulo de Pascal. Distribuição de freqüência. Preveja o resultado 774



PLANO DA OBRA

"INPUT" é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

COMPLETE SUA COLEÇÃO

Exemplares atrasados, até seis meses após o encerramento da coleção, poderão ser comprados, a preços atualizados, da seguinte forma: 1. PESSOALMENTE — Por meio de seu jornaleiro ou dirigindo-se ao distribuidor local, cujo endereço poderá ser facilmente conseguido junto a qualquer jornaleiro de sua cidade. Em **São Paulo**, os endereços são: rua Brigadeiro Tobias, 773, Centro; av. Industrial, 117, Santo André; e no **Rio de Janeiro**: rua da Passagem, 93, Botafogo. 2. POR CARTA — Poderão ser solicitados exemplares atrasados também por carta, que deve ser enviada para DINAP — Distribuidora Nacional de Publicações — Números Atrasados — Estrada Velha de Osasco, 132, Jardim Teresa — CEP 06000 — Osasco — SP. Não envie pagamento antecipado. O atendimento será feito pelo reembolso postal e o pagamento, incluindo as despesas postais, deverá ser efetuado ao se retirar a encomenda na agência do Correio. 3. POR TELEX — Utilize o nº (011) 33 670 DNAP.

Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações, Lda. — Qta. Pau Varais, Azinhaga de Fetais — 2 685, Camarate — Lisboa; Apartado 57 — Telex 43 069 JARLIS P.

Atenção: Após seis meses do encerramento da coleção, os pedidos serão atendidos dependendo da disponibilidade do estoque.

Obs.: Quando pedir livros, mencione sempre título e/ou autor da obra, além do número da edição.

COLABORE CONOSCO

Encaminhe seus comentários, críticas, sugestões ou reclamações ao Serviço de Atendimento ao Leitor — Caixa Postal 9442, São Paulo — SP.

Editor
VICTOR CIVITA

REDAÇÃO
Diretora Editorial: Iara Rodrigues

Editor Executivo: Antonio José Filho

Editor Chefe: Paulo de Almeida
Editor de Texto: Cláudio A. V. Cavalcanti
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Ailton Oliveira Lopes, Dilvacy M. Santos, Grace Alonso Arruda, José Maria de Oliveira, Monica Lenardon Corradi
Secretária de Redação/Coordenadora: Stefania Crema
Secretários de Redação: Beatriz Hagström, José Benedito de Oliveira Damião, Maria de Lourdes Carvalho, Marisa Soares de Andrade, Mauro de Queiroz

COLABORADORES
Consultor Editorial Responsável: Dr. Renato M. E. Sabbatini (Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas)
Execução Editorial: DATAQUEST Assessoria em Informática Ltda., Campinas, SP
Tradução: Reinaldo Cúrcio
Tradução, adaptação, programação e redação: Abílio Pedro Neto, Aluísio J. Dornellas de Barros, Marcelo R. Pires Therezo, Raul Neder Porrelli
Coordenação Geral: Rejane Felizatti Sabbatini
Editora de Texto: Ana Lúcia B. de Lucena
Assistente de Arte: Dagmar Bastos Sampaio

COMERCIAL
Diretor Comercial: Roberto Martins Silveira
Gerente Comercial: Flávio Maculan
Gerente de Circulação: Denise Maria Mozol

PRODUÇÃO
Gerente de Produção: João Stungis
Coordenador de Impressão: Atilio Roberto Bonon

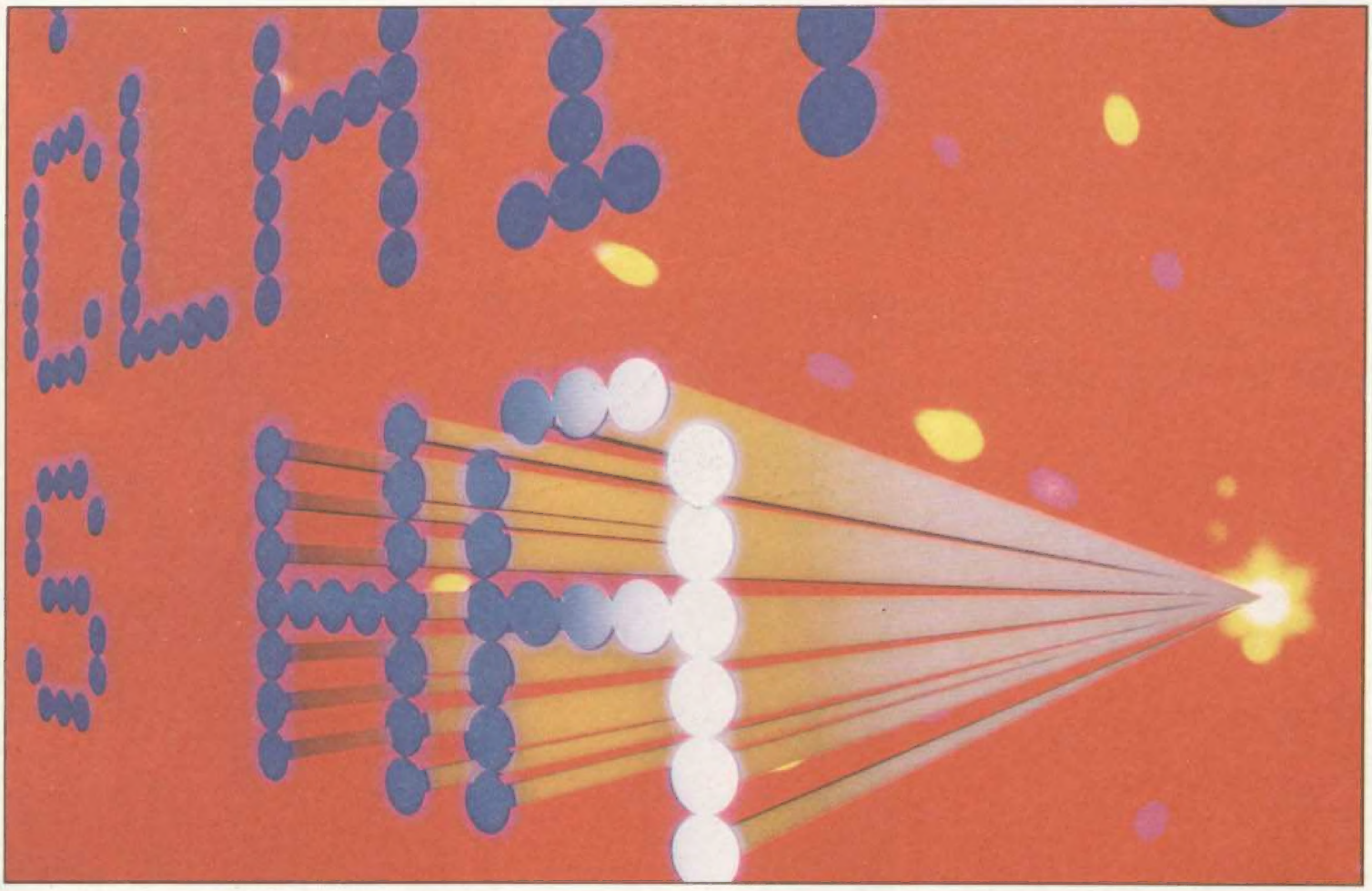
Preparador de Texto/Coordenador: Eliel Silveira Cunha
Preparadores de Texto: Alzira Moreira Braz, Ana Maria Dilguerian, Karina Ap. V. Grechi, Levon Yacubian, Luciano Tasca, Maria Teresa Galluzzi, Maria Teresa Martins Lopes, Paulo Felipe Mendrone
Revisor/Coordenador: José Maria de Assis
Revisoras: Conceição Aparecida Gabriel, Isabel Leite de Camargo, Ligia Aparecida Ricetto, Maria de Fátima Cardoso, Nair Lúcia de Brito
Paste-up: Anastase Potaris, Balduino F. Leite, Edson Donato

© Marshall Cavendish Limited, 1984/85.
© Editora Nova Cultural Ltda., São Paulo, Brasil, 1986.

Edição organizada pela Editora Nova Cultural Ltda.
Av. Brigadeiro Faria Lima, nº 2000 - 3º andar
CEP 01452 - São Paulo - SP - Brasil
(Artigo 15 da Lei 5 988, de 14/12/1973).
Esta obra foi composta na AM Produções Gráficas Ltda. e impressa na Divisão Gráfica da Editora Abril S.A.

AS INSTRUÇÕES DO JOGO

- MAIS TABELAS DE DADOS EM ASCII
- ESCREVA NA TELA
- COMO CHAMAR SUB-ROTINAS DA ROM



Impressas na tela, as informações contidas no programa deste artigo permitirão que ajudemos o infortunado Willie em sua incansável busca do lanche roubado.

A primeira coisa a fazer é colocar o título na tela. Em seguida, são fornecidas as instruções. Como estamos programando e conhecemos cada detalhe envolvido, às vezes esquecemos que outras pessoas não têm a menor idéia dos objetivos do jogo e não sabem como jogar. Assim, é necessário colocar na tela toda informação relevante, de maneira que o usuário não seja obrigado a parar o jogo a cada momento para ler o folheto de instruções. As informações devem ser bem claras e objetivas. Ao mesmo tempo, é conveniente "temperar" o texto com algum enredo, já que um videogame deve apelar à imaginação do jogador e não simplesmente testar seus reflexos.

S

Mais uma vez, utilizaremos um programa em BASIC para colocar as instruções — letra por letra — na memória, criando uma tabela de códigos ASCII.

Para que a tabela sirva aos nossos propósitos, é importante que o número de códigos de caracteres transferidos para a memória seja exato — afinal, eles vão ser lidos pelo programa em código. Como o leitor pode ter problemas para descobrir o número de espaços em cada linha, vamos conferir: no final da linha 100, há um espaço; no final da linha 120, três; na 140, três; na 160, catorze; na 180, um; na 240, dois; na 300, três, e na 320, 26 espaços em branco. Nas linhas 330 e 340 há quatro espaços entre as teclas e o hífen, e um espaço entre o hífen e a função da tecla. No final da linha 330 temos dezenove espaços e no final da 340, dezenove espaços também. Todos os demais claros na listagem correspondem a um caractere de espaço apenas. Outra maneira de conferir os dados consiste em contar exatamente 32 caracteres em cada linha **DATA**, exceto na 320, que tem 34 caracteres.

Antes de passar à execução, proteja o topo da memória com **CLEAR 57434**.

Quando executarmos o programa, será criada uma tabela com instruções. Ao

final do programa, o laço **FOR...NEXT** entre as linhas 500 e 520 projeta na tela uma imagem da porção da memória que contém a tabela.

```
 10 LET x=57480
 20 FOR n=1 TO 16
 30 READ a$: FOR o=1 TO LEN a$
 : POKE x,CODE (a$(o TO o)):
   LET x=x+1: NEXT o
 40 NEXT n
100 DATA " Apos um pequeno pas
seio,Willie "
120 DATA "volta e descobre que
 um bando   "
140 DATA "de cabritos espalhou
 todo seu   "
160 DATA "lanche na encosta.
           "
180 DATA "Agora ele deve subir
 ao topo da "
200 DATA "montanha e recuperar
 suas coisas"
220 DATA "enfrentando uma aval
anche,cobras"
240 DATA "e podendo ate cair n
um buraco.  "
260 DATA "Para piorar a situac
ao a mare   "
280 DATA "esta subindo e ele c
orre o risco"
290 DATA "de se afogar. Para a
judar Willie"
300 DATA "leia as instrucoes e
 aperte a   "
320 DATA "tecla 'S'
           "
330 DATA "N    - Correr
           "
340 DATA "M    - Saltar
           "
360 DATA "  ou Ambos"
500 FOR n=57435 TO 58000
510 PRINT CHR$ PEEK n;
520 NEXT n
```

Os dados que este programa coloca na memória usando **POKE** são utilizados pela seguinte rotina em Assembly:

```
 10 REM org 58104
 20 REM call cl
 30 REM ld ix,57480
 40 REM ld hl,32
 50 REM ld a,7
 60 REM ld b,255
 70 REM call me
 80 REM ld b,138
 90 REM call me
100 REM ld de,39
110 REM add hl,de
120 REM ld b,100
130 REM ld a,70
140 REM call me
150 REM ktt ld a,253
160 REM in a,254
170 REM bit 1,a
180 REM jr nz,ktt
190 REM ret
200 REM org 58192
210 REM cl *
220 REM org 58155
230 REM me *
```

Esses dois programas — ou o seu produto, criado após a execução ou a montagem pelo Assembler — devem ser gravados da mesma maneira que a primeirar parte do videogame *Avalanche* (veja artigo publicado à página 748).

Para que tal rotina funcione adequadamente, essa parte do jogo e a tabela criada pelo primeiro programa BASIC também devem estar na memória. Isso exige que elas sejam gravadas em código; caso contrário, terão de ser montadas novamente.

Para gravar e ler código de máquina em fita cassete — o que inclui os códigos da tabela ASCII —, o usuário do Spectrum tem à sua disposição comandos **SAVE** e **LOAD** especiais.

Para gravar os códigos de uma porção da memória, digite:

```
SAVE "nome" CODE endereço inici
al,nº de bytes
```

Para recuperar os mesmos códigos da fita cassete, use:

```
LOAD "nome" CODE endereço inici
al
```

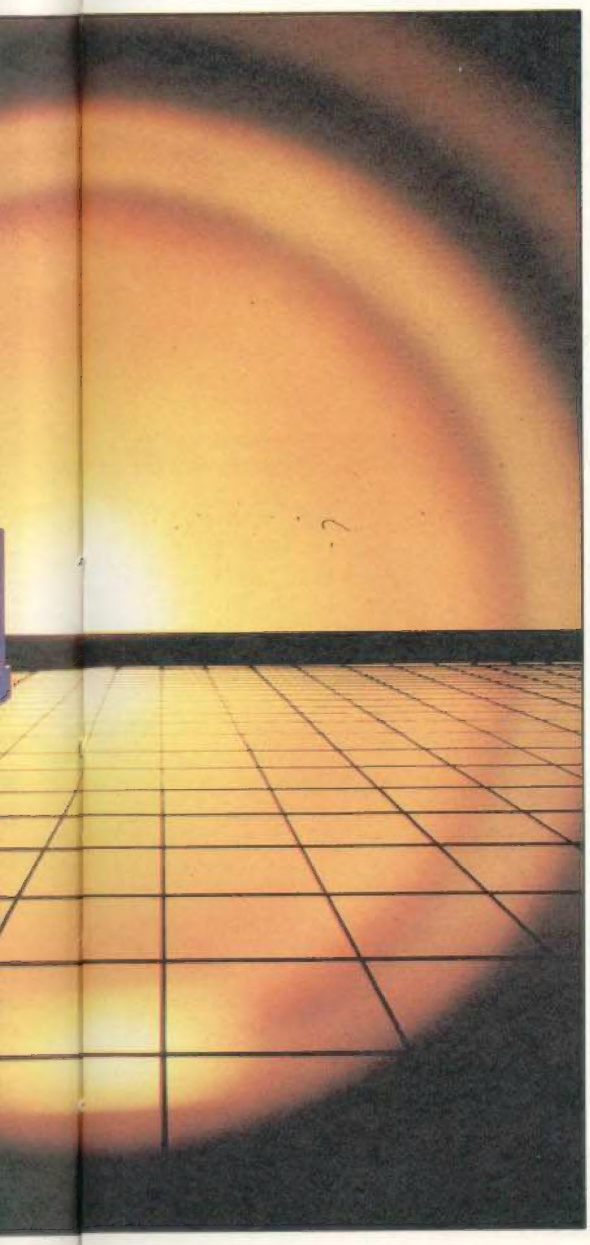
O **endereço inicial** pode ser omitido se for igual ao do momento da gravação. Os códigos das diversas partes do videogame *Avalanche* também devem ser gravados; caso isso não aconteça, o usuário gastará cada vez mais tempo na montagem, adicionando novas rotinas e tabelas. Os endereços iniciais e finais devem ser anotados. O próprio Assembler dá essa informação durante a montagem.

Também é bom registrar os endereços dos rótulos, para utilizá-los em instruções com asteriscos (cálculos de saltos e desvios).

ESCREVA AS INSTRUÇÕES NA TELA

A rotina **cl** — que está na primeira parte do programa — é chamada para fazer a limpeza da tela. Já a rotina **me** será usada novamente para escrevermos na tela; para isso, temos que colocar valores adequados em certos registros, que controlam as características do texto a ser impresso.

Para começar, alocamos no registro IX o endereço inicial da nova tabela de códigos ASCII criada pelo programa BASIC. A posição da tela em que começaremos a escrever deve ser colocada no par de registros HL; a cor das letras, em A, e o comprimento do texto, em B. Para imprimir os vários segmentos do texto, é preciso chamar diversas vezes a rotina **me**. Antes de cada chamada, porém, devem ser distribuídos os parâmetros relacionados — posição inicial na tela e número e cor das letras — nos registros adequados.



Quando chamarmos **me** pela primeira vez, B estará contendo o valor 255, que corresponde ao comprimento do texto. Na segunda vez, B estará contendo 138. A rotina deve ser chamada duas vezes porque 255 é o maior número que o registro B, de oito bits, pode abrigar. 138 é o restante do texto, que não pode ser impresso de uma só vez por **me**. Não é preciso definir novamente a posição de impressão na tela — HL —, já que a segunda parte dos dados segue imediatamente a primeira e o conteúdo desse par é atualizado pela própria rotina de impressão (veja a rotina **me** na primeira parte do programa).

Quando quisermos imprimir a terceira parte do texto, contudo, precisaremos usar uma nova posição na tela. Isto é feito para criar um espaço entre o texto inicial — que conta os azares de Willie — e as instruções sobre o uso do teclado, que orientam sobre quais teclas apertar para saltar e/ou correr. Existem duas maneiras de criar esse espaço: colocando uma série de códigos de espaço na tabela ASCII, ou dando um novo valor a HL. Procuramos ilustrar as duas técnicas no programa. Se consultarmos a listagem do programa BASIC que criou a tabela de dados, veremos que existem 26 espaços em branco na linha 320 (naturalmente eles foram transferidos para a tabela). Olhando a listagem Assembly, verificaremos que o número 39 foi colocado em DE pelo comando **ld de,39**. Depois disso, o conteúdo de DE foi somado ao de HL. O par HL funciona como um acumulador de dezesseis bits, de forma que o resultado da operação permanece em HL, que é o local apropriado para o endereço de impressão na tela quando chamamos novamente a rotina **me**.

O restante do texto tem cem caracteres de comprimento e é impresso na cor amarelo brilhante — código 70 — e não mais em branco — código de cor 7 —, como as linhas iniciais.

ESPERE PELO TIRO DE LARGADA

As instruções permanecem no vídeo até que pressionemos a tecla S, dando início ao jogo propriamente dito. Para tornar isso possível, usamos uma rotina que detecta mudanças no teclado por meio do comando **in**.

Esse comando permite o que chamamos de "varredura do teclado", de um modo idêntico ao usado no programa *Trace*, do artigo *Rastreamento no Spectrum* (página 381). Naquela ocasião, queríamos detectar as teclas **<BREAK>** e **<SYMBOL SHIFT>**; agora, nossas atenções estão na tecla S. É o valor colocado em A, antes da execução do comando **in**, que determina a porção do teclado a ser verificada. Assim, **ld a,253** faz com que a linha **in a,254** verifique o conjunto de teclas ao qual S pertence.

O valor recebido através da porta 254 é colocado em A. Se o bit 1 desse número estiver "ligado" é porque S foi pressionada. Quando isso acontece, o resultado da operação **bit 1,a** é zero, ativando o indicador de zero.

O processador ficará girando, preso no laço rotulado como **ktt**, até que a tecla S seja pressionada, ativando o indicador de zero e evitando o salto **jr nz, ktt**, já que a condição **nz** — resultado da última operação **não** é **zero** — não foi satisfeita. Normalmente, o programa prosseguiria; por enquanto, porém, uma instrução **ret** provocará o retorno ao BASIC, terminando assim esta segunda parte de *Avalanche*.

Note que os rótulos correspondentes a rotinas montadas em outras listagens devem ter seu endereço definido por **org**, seguido do rótulo e de um asterisco. Esse procedimento permite ao Assembler calcular os saltos para tais rótulos (**cl** e **me**, neste caso). O asterisco impede que a posição de memória correspondente seja alterada.

Se chamarmos o programa tal como foi estruturado até aqui, ele escreverá a tela de instruções logo após a impressão da tela do título, e aguardará que pressionemos a tecla S. A instrução **ret** do final do primeiro programa foi apagada pela montagem da segunda parte, de modo a fazer das duas listagens um todo único e coerente.

T

Assim como no exemplo precedente, utilizaremos aqui um programa escrito em BASIC para colocar — letra por letra — as instruções na memória, criando uma tabela de códigos ASCII. Adicione as próximas linhas ao programa do artigo anterior. Antes de executá-lo, lembre-se de proteger o topo da memória com **CLEAR 200, 17999**.

```
20 FOR I=1 TO 4
70 NEXT A,I
80 DATA "avalanchecriacao: a.do
```

```
e programa: s. kellaway e g. he
dley"
85 DATA "  depois de um pequeno
 passeio    willie descobre que
 cabritos    monteses espalhara
m todo seu    lanche pela encos
ta.            agora ele tem de
 subir ao topo "
90 DATA " da montanha pegar sua
s coisas.  no caminho ele sera
ameacado    por pedras, cobras
e buracos.   para piorar a situ
acao, a mare  esta subindo, pod
endo afoga-lo. para ajudar will
ie leia as       instrucoes e ap
erte 's' para    jogar.  "
100 DATA "m - correrm - saltarm
n - ambos"
```

Para que a tabela sirva aos nossos propósitos, é importante que o número de códigos de caracteres transferidos para a memória seja exato (afinal, eles vão ser lidos pelo programa em código). Como o leitor pode ter problemas para descobrir o número de espaços em cada linha, é conveniente conferi-los: no final da linha 85, aparecem conjuntos de espaços com os seguintes tamanhos: 2, 1, 4, 1, 4, 4 e 12, segundo a ordem. Na linha 90, a seqüência é a seguinte: 1, 2, 4, 3, 2, 1, 6, 4 e 2. Na linha 100, os espaços têm apenas um caractere em branco.

A seguir, apresentamos o programa em código que utiliza a tabela ASCII criada pela listagem anterior. Para montá-lo na memória, recorra ao Assembler de *INPUT*. Antes disso, porém, use os comandos **POKE** que aumentam a memória disponível e proteja o topo da memória com **CLEAR 200,18999**.

```
10   ORG 19054
20   LDX #1024
30   LDY #17058
40   CLRB
50   JSR LPRINT
60   LDB #137
70   JSR LPRINT
80   LEAX 24,X
90   LDB #10
100   JSR LPRINT
110   LEAX 22,X
120   LDB #10
130   JSR LPRINT
140   LEAX 22,X
150   LDB #10
160   JSR LPRINT
170   KEY JSR 41409
180   CMPA #83
190   BNE KEY
200   RTS
210   LPRINT EQU 19174
```

Esses dois programas — ou o seu produto, criado depois de ter sido executado (ou montado) pelo Assembler — devem ser gravados da mesma maneira que a primeira parte do videogame *Avalanche* (os endereços iniciais e finais devem ser calculados e usados no comando de gravação dos códigos). Para que a rotina acima funcione adequadamente, a primeira parte do jogo e a tabela criada pelo BASIC completo devem estar na memória também.

Faça a gravação dos códigos correspondentes às tabelas e aos programas, empregando o comando **CSAVEM**, que tem a seguinte sintaxe:

```
CSAVEM "nome",endereço inicial,
endereço final
```

ESCREVA AS INSTRUÇÕES NA TELA

Esse programa usa a mesma sub-rotina **LPRINT** que a rotina da página do título empregou para escrever na tela. Note que a tabela ASCII criada pelo programa BASIC é uma continuação da que continha os códigos da página do título, de forma que o programa está utilizando a mesma tabela ASCII.

Toda vez que chamarmos **LPRINT**, devemos colocar nos registros apropriados os parâmetros do texto a ser impresso. Assim, X deve conter a posição da tela onde desejamos iniciar a impressão, B, o comprimento do texto, e Y, a posição do início do texto na tabela.

Se olharmos para a listagem Assembly da sub-rotina **LPRINT** — apresentada na primeira parte —, veremos que o registro Y tem seu valor aumentado na medida em que a tabela ASCII vai sendo "lida" pelo programa. Desse modo, o programa principal não precisará corrigir o valor de Y toda vez que a rotina for chamada; na verdade, Y é aumentado, gradativamente, indicando sempre, linha após linha, o início da porção apropriada da tabela.

Definida a origem do programa, o endereço inicial da tela é colocado em X por **LDX #1024** e o endereço inicial do texto das instruções é posicionado em Y por **LDY #17060**. A instrução **CLRB** limpa o conteúdo de B, sendo um modo rápido de colocar zero nesse registro. O valor zero funciona como se fosse 256; assim, quando a rotina **LPRINT** for chamada por **JSR LPRINT**, serão impressos 256 caracteres.

Ao chamarmos **LPRINT** pela primeira vez, B estará contendo o valor 256, que corresponde ao comprimento do texto. Na segunda vez, B estará contendo 137. A rotina será chamada duas vezes, pois 256 é o maior número que o registro B, de oito bits, pode conter. 138 é o resto do texto que não pode ser impresso todo de uma vez por **LPRINT**.

As instruções **LEAX** somam valores adequados ao conteúdo de X para que a posição de impressão na tela corresponda ao início da linha seguinte; é por isso que não há espaços entre as palavras que ficam no final e no início de linhas adjacentes. A sub-rotina **LPRINT** é chamada então mais quatro vezes para escrever as linhas que orientam as funções das teclas. A cada chamada, o valor de B é acertado por uma instrução **LDB**; o valor de X é ajustado por um **LEAX** (Y é acertado pela própria rotina **LPRINT**).

A sub-rotina **KEY** aguarda que a tecla S seja acionada para que o jogo tenha início. Para verificar o teclado, o programa utiliza uma sub-rotina da ROM. Assim, a instrução **JSR 41409** salta para essa sub-rotina, que verifica qual tecla está sendo pressionada.

Se alguma tecla estiver sendo pressionada, o processador retornará da sub-rotina trazendo o código da tecla correspondente dentro do acumulador. A instrução **CMPA #83** compara então o código da letra S — 83 — com o conteúdo do registro A. Se S não tiver sido acionada, a condição de **BNE KEY** ("salta, se não for igual") estará satisfeita e o microprocessador será enviado novamente ao rótulo **KEY**. O processo se repetirá até que seja pressionada a tecla S. Nesse caso, a condição de **BNE** não estará satisfeita e o programa seguirá seu curso até ser interrompido por uma **RET**. O rótulo correspondente a **LPRINT** deve ter o endereço especificado por uma instrução **EQU**; assim, o Assembler poderá calcular os saltos para aquela sub-rotina.

O Assembly listado a seguir cria a tela de instruções ao jogador. Para fazer isso, ele utiliza dados de uma tabela de códigos ASCII criada por um programa BASIC; nele incluímos uma rotina de "varredura" do teclado, que aguarda que pressionemos a tecla S:

```
10    org -12233
20    ld de,0
30    ld hl,-13958
40    ld bc,960
50    call 92
60    ktt call 159
70    cp 83
80    jr nz,ktt
90    ret
100    end
```

Para montar esse programa, utilize nosso Assembler. Grave o programa-fonte e o programa-objeto da mesma forma que na primeira parte do videogame.

Como das vezes anteriores, precisaremos de um BASIC para colocar na memória os códigos das letras que compõem a tela de instruções. Digite-o, mas não o execute ainda.

```
10 SCREEN 0:CLEAR 200,-14000:KE
Y OFF
20 FOR I=1 TO 22: READ AS: PRIN
T TAB(1);AS: NEXT
30 FOR I=0 TO 959
40 POKE -13958+I,VPEEK (BASE(0)
+I)
50 NEXT
60 CLS
70 DEFUSR1=-12288
80 A=USR1(0)
90 CLS
100 END
200 DATA "     Após um pequeno p
asseio, Willie"
210 DATA "volta ao local onde p
retendia fazer"
220 DATA "seu pique-nique e des
cobre que um"
230 DATA "bando de cabritos mon
teses espalhou"
240 DATA "todo seu lanche na en
costa."
245 DATA
250 DATA "     Agora ele deve su
bir ao topo da "
260 DATA "montanha para recuper
ar suas coisas, "
270 DATA "enfrentando uma avala
nche, cobras"
280 DATA "venenosas e correndo
o risco de cair"
290 DATA "num buraco."
295 DATA
300 DATA "     Para piorar a sit
uação, a maré"
310 DATA "está subindo e ele po
de se afogar."
320 DATA "Se você quer ajudar W
illie, leia as"
330 DATA "instruções e pression
e a tecla 'S'."
335 DATA:DATA
340 DATA "Use:"
350 DATA "      N       para
Correr"
360 DATA "      M       para
Saltar"
370 DATA "      Ambos   para um
Salto Diagonal"
```

As linhas 70 e 80 tentam executar o programa do jogo como foi publicado até o momento. Se a primeira parte dele não estiver na memória ainda, a execução do comando da linha 80 vai apagar toda a memória.

Esse programa utiliza o comando **READ** para obter as frases da tela de instrução nas linhas **DATA** do final da listagem. Depois de escrita, a tela é totalmente transferida para a memória pelo laço entre as linhas 30 e 50. A tabela ASCII é criada assim: o comando **VPEEK** obtém os códigos diretamente na memória de vídeo e o comando **POKE** os coloca na RAM.

Para que a tela mantenha a tabulação original é bom conferir: no início das linhas 200, 250 e 300, há quatro espaços em branco. Nas linhas 350 e 360, cada vazio no texto contém cinco espaços. Uma vantagem dessa técnica de transferência da tela para a memória é que você pode criar sua própria tela de instruções se não estiver satisfeito com o aspecto da nossa.

COMO MONTAR O PROGRAMA INTEIRO

Eis algumas explicações adicionais sobre como montar o programa.

A primeira coisa a fazer é carregar o Assembler na memória. Depois de tomar as providências necessárias para proteger a memória na linha 5000, monte o programa-fonte do artigo anterior. Durante a montagem é interessante anotar o endereço inicial, o endereço final e os endereços dos rótulos.

Terminada a montagem, a rotina em código estará na memória. Quem ainda não a gravou em fita deve fazê-lo por intermédio do comando:

```
BSAVE "CAS:AVAL1",-12288,-12233
```

A seguir, o Assembly deve ser digitado, gravado e montado (anote os endereços principais durante o processo de montagem). A rotina em código pode então ser gravada da mesma maneira.

Depois, digite **NEW**, apagando o Assembler da memória, e carregue o BASIC que cria a tabela ASCII da tela-título, listado no artigo anterior. Quem já gravou a tabela correspondente com o comando **BSAVE** pode carregá-la da fita com **BLOAD**, mas terá que proteger o topo da memória antes. Quem não fez isso ainda deve executar o programa. As linhas finais deste último testam a rotina em código; com a sua execução, a página-título surge na tela; após uma pequena pausa, esta se enche de caracteres aleatórios. Isso acontece porque a rotina tenta escrever a tela de instruções, mas a tabela necessária não se encontra ainda na memória. Para continuar, aperte a tecla S e limpe a tela.

Finalmente, carregue o programa BASIC apresentado linhas atrás e execute-o. As linhas finais testam a rotina completa, que escreve o título, provoca uma pausa, apaga a tela, escreve as instruções e aguarda que a tecla S seja pressionada. A tabela ASCII completa pode ser então gravada com:

```
BSAVE "CAS:ASCII",-14000,-12998
```

É aconselhável gravar os códigos das rotinas e tabelas usando o comando BSAVE. Senão, teremos que gastar um tempo cada vez maior para a montagem. Após gravar os códigos, proteja a memória, digitando a instrução.

```
CLEAR 200,-14000
```

(proteção feita antes pelos programas BASIC). Posicione a fita nas rotinas em código e carregue-as com:

```
BLOAD "CAS:"
```

Finalmente, posicione a fita nos códigos da tabela ASCII e carregue-a usando o mesmo comando **BLOAD**. Para executar o programa, digite:

```
DEFUSR=-12288:A = USR(0)
```

ESCREVA NA TELA

Esse programa recorre à mesma sub-rotina da ROM utilizada pela rotina da página do título para escrever na tela. A tabela ASCII criada pelo programa BASIC é uma continuação da que continha os códigos da página do título, de forma que o programa está utilizando a mesma tabela ASCII.

Toda vez que chamarmos essa sub-rotina, devemos colocar nos registros apropriados os parâmetros do texto a ser impresso. Assim, DE deve conter a posição da tela onde desejamos iniciar a impressão do texto; BC, o comprimento do texto; e HL, a posição do início do texto na tabela ASCII.

A sub-rotina **ktt** aguarda que a tecla S seja pressionada para que o jogo comece. O programa verifica o teclado utilizando uma sub-rotina da **ROM**. Assim, a instrução **call 159** salta para essa sub-rotina, que espera até uma tecla ser pressionada.

Se alguma tecla for pressionada, o processador retornará da sub-rotina trazendo o código da tecla correspondente dentro do acumulador. A instrução **cp 83** compara então o código da letra 2 — 83 — com o conteúdo de A. Se S não tiver sido pressionada, a condição do comando **jr nz,ktt** ("salta, se o último resultado não for zero") estará satisfeita e o microprocessador será novamente enviado para o rótulo **ktt**. O processo se repetirá até que seja pressionada a tecla S. Nesse caso, a condição de **nz** não será satisfeita e o programa prosseguirá seu curso, até encontrar uma instrução **ret**.

Ao executarmos o programa, veremos aparecer um cursor no canto superior esquerdo da tela de instruções. Ele é colocado ali pela rotina da ROM que verifica o teclado. Para tirá-lo dessa posição, teríamos que escrever nossa própria rotina para detectar o toque na tecla S. Depois que a tecla for pressionada e o programa terminar, o texto permanecerá na tela. Sabemos que a rotina terminou porque surge a mensagem "OK" no meio do texto.

TUDO QUE SOBE, DESCE...

Um objeto lançado para o alto descreve uma curva parabólica. Isaac Newton (1642-1727) demonstrou que, se a altura de lançamento for muito grande, o objeto cairá fora da Terra.

Segundo Galileu Galilei (1564-1642), todo objeto lançado sobre a superfície da Terra sofre a ação de duas forças: a que o impulsiona para a frente (ou para o alto) e a que o atrai para o planeta (força de gravidade). A interação dessas duas forças leva o corpo a descrever uma curva parabólica. O desenho desse tipo de trajetória torna-se muito simples quando se trabalha com um computador.

Um objeto lançado para cima, por exemplo, distancia-se — a princípio, rapidamente — das superfície da Terra sob a ação do impulso inicial, perdendo velocidade por efeito da gravidade. Este artigo mostra como programar e simular o movimento de projéteis.

Muitos jogos de combate são ambientados no vácuo do espaço sideral porque lá se pode ignorar a ação da força gravitacional e da resistência do ar no cálculo do movimento de naves e mísseis. Isso não quer dizer que não exista nenhuma gravidade no espaço; existe, mas é tão pequena que pode ser desprezada para efeitos práticos.

Em contrapartida, batalhas simuladas na superfície terrestre têm que levar em conta a ação da gravidade e, freqüentemente, a resistência do ar e o efeito dos ventos. Além de jogos de guerra, podem surgir várias outras circunstâncias em que o conhecimento de como se dá o movimento de projéteis seja essencial para um efeito realístico. Entre elas estão muitos esportes e simulação de provas de atletismo, como lançamento de dardo e de disco, mergulho e todo o tipo de salto.

Os projéteis citados (sejam pessoas ou objetos) têm uma coisa em comum: eles se movem segundo uma trajetória que pertence a um grupo de curvas conhecidas como parábolas. Escrever um programa para simular movimento em trajetória parabólica não é difícil, requerendo apenas a compreensão das forças que atuam sobre o objeto e o emprego de matemática elementar.

Saber, por exemplo, que o movimento executado por uma bola ao ser chutada resulta da combinação de movimento em duas direções já fornece subsídios suficientes para resolver o problema da programação. Uma dessas direções é paralela ao eixo horizontal ou eixo X de um sistema cartesiano. A outra é vertical, paralela ao eixo Y. Nesta lição, estabeleceremos como premissa que o objeto se move com velocidade constante na direção horizontal. Na realidade, o objeto seria desacelerado pela resistência do ar, mas este é um detalhe que costuma ser ignorado, a não ser em trabalhos de grande precisão.

MOVIMENTO HORIZONTAL

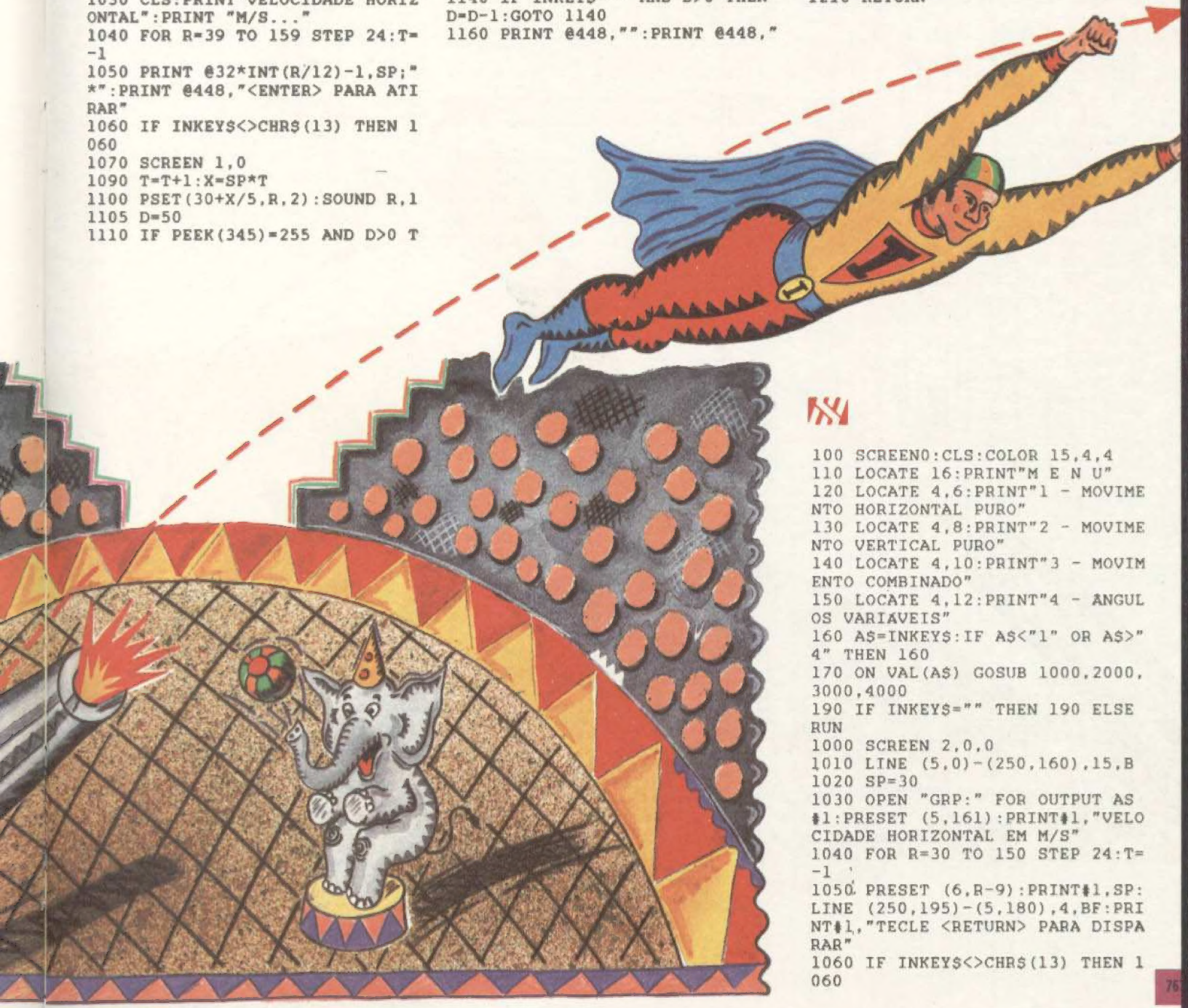
Digite esse primeiro programa para simular o movimento horizontal, e cuidado para não confundir a letra I com o número 1. Todos os programas são para o BASIC padrão da máquina.

```
100 BORDER 7: PAPER 7: INK 0:
CLS
105 POKE 23658,8
110 PRINT  INVERSE 1;AT 2,12;"
 MENU "
120 PRINT AT 6,0;"1-MOVIMENTO
PURAMENTE HORIZONTAL"
130 PRINT AT 8,0;"2-MOVIMENTO
PURAMENTE VERTICAL"
140 PRINT AT 10,0;"3- MOVIMENT
O MISTO"
150 PRINT AT 12,0;"4- ELEVACOE
S"
200 LET I$=INKEY$: IF I$=""
THEN  GOTO 200
205 IF I$<"1" OR I$>"4" THEN
 GOTO 200
210 GOSUB VAL I$*1000
220 RUN
1000 CLS
1020 LET SP=30
1030 PRINT "VELOC. HORIZONTAL M
/S..."
1040 FOR R=124 TO 28 STEP -16
1045 LET T=0
1050 PRINT AT 21-(R/8),0;SP
1060 INPUT "<ENTER> PARA ATIRAR
", LINE Z$
1090 LET X=SP*T: LET T=T+1
1100 PLOT 30+X,R: SOUND .1,R/4
1110 PAUSE 10
1120 IF 30+SP*T<250 THEN  GOTO
1090
1150 IF R=28 THEN  GOTO 1200
1160 INPUT "NOVA VELOCIDADE (0
SAI)", LINE I$
1165 LET SP=VAL I$
1170 IF SP<0 OR SP>1000 THEN  G
OTO 1160
1190 IF SP=0 THEN  LET R=28
1200 NEXT R
1210 RETURN
```

```
100 CLS:PMODE 3
110 PRINT @13,"menu"
120 PRINT @128,"1-MOVIMENTO PUR
AMENTE HORIZONTAL"
130 PRINT @192,"2-MOVIMENTO PUR
AMENTE VERTICAL"
140 PRINT @256,"3-MOVIMENTO MIS
TO"
150 PRINT @320,"4-ELEVACOES"
160 A$=INKEY$:IF A$<"1" OR A$>"
4" THEN 160
170 ON VAL(A$) GOSUB 1000,2000,
3000,4000
```

- COMO DESENHAR A TRAJETÓRIA DE PROJÉTEIS
- * EFEITOS DE DIFERENTES CAMPOS GRAVITACIONAIS
- COMO SIMULAR PARÁBOLAS
- FUNCIONAMENTO DO PROGRAMA
- MOVIMENTO VERTICAL
- MOVIMENTO HORIZONTAL
- COMO UNIR AS ROTINAS
- MUDE O ÂNGULO

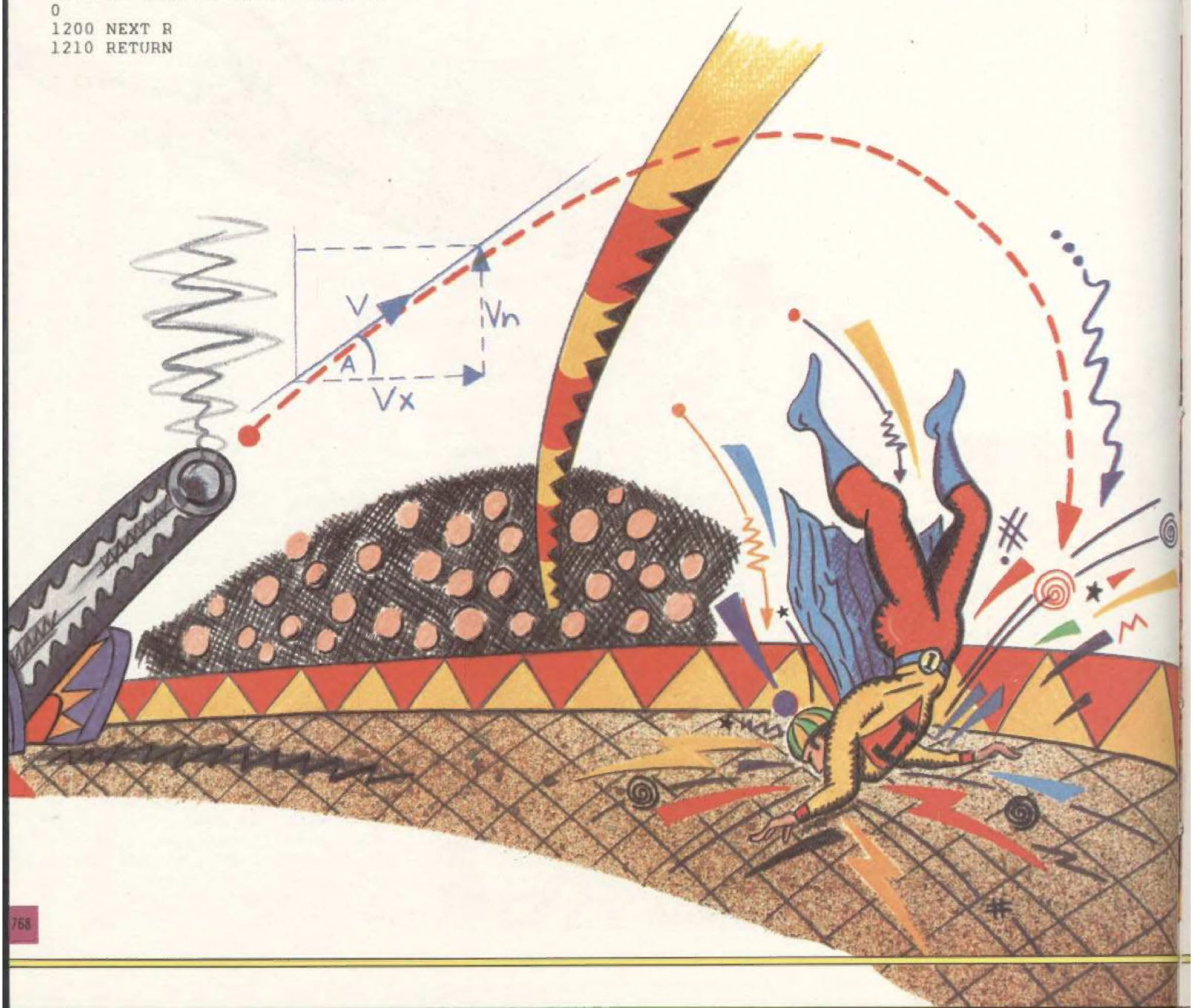
```
180 A$=INKEY$
190 IF INKEY$="" AND PEEK(65314
)<>7 THEN 190 ELSE RUN
1000 PCLS
1010 LINE(0,0)-(255,191),PSET,B
1020 SP=30
1030 CLS:PRINT"VELOCIDADE HORIZ
ONTAL":PRINT "M/S..."
1040 FOR R=39 TO 159 STEP 24:T=
-1
1050 PRINT @32*INT(R/12)-1,SP;"
*":PRINT @448,"<ENTER> PARA ATI
RAR"
1060 IF INKEY$<>CHR$(13) THEN 1
060
1070 SCREEN 1,0
1090 T=T+1:X=SP*T
1100 PSET(30+X/5,R,2):SOUND R,1
1105 D=50
1110 IF PEEK(345)=255 AND D>0 T
HEN D=D-1:GOTO 1110
1120 IF SP*(T+1)/5<223 THEN 109
0
1130 IF R>150 THEN 1200
1135 D=200
1139 A$=INKEY$
1140 IF INKEY$="" AND D>0 THEN
D=D-1:GOTO 1140
1160 PRINT @448,"":PRINT @448,"
NOVA VELOCIDADE (0 SAI):";:INPU
T SP
1170 IF SP<0 OR SP>999 THEN 116
0
1190 IF SP=0 THEN R=160
1200 NEXT
1210 RETURN
```



```
100 SCREEN0:CLS:COLOR 15,4,4
110 LOCATE 16:PRINT"M E N U"
120 LOCATE 4,6:PRINT"1 - MOVIME
NTO HORIZONTAL PURO"
130 LOCATE 4,8:PRINT"2 - MOVIME
NTO VERTICAL PURO"
140 LOCATE 4,10:PRINT"3 - MOVIM
ENTO COMBINADO"
150 LOCATE 4,12:PRINT"4 - ANGUL
OS VARIAVEIS"
160 A$=INKEY$:IF A$<"1" OR A$>"
4" THEN 160
170 ON VAL(A$) GOSUB 1000,2000,
3000,4000
190 IF INKEY$="" THEN 190 ELSE
RUN
1000 SCREEN 2,0,0
1010 LINE (5,0)-(250,160),15,B
1020 SP=30
1030 OPEN "GRP:" FOR OUTPUT AS
#1:PRESET (5,161):PRINT#1,"VELO
CIDADE HORIZONTAL EM M/S"
1040 FOR R=30 TO 150 STEP 24:T=
-1
1050 PRESET (6,R-9):PRINT#1,SP:
LINE (250,195)-(5,180),4,BF:PRI
NT#1,"TECLE <RETURN> PARA DISPA
RAR"
1060 IF INKEY$<>CHR$(13) THEN 1
060
```

```
1090 T=T+1:X=SP*T
1100 PSET (30+X/5,R),15:PLAY"N1
9L19"
1110 FOR I=1 TO 100:IF INKEY$ <
>CHR$(32) THEN NEXT I
1120 IF SP*(T+1)/5+30<240 THEN
1090
1130 IF R>145 THEN 1200
1140 FOR I=1 TO 200:IF INKEY$<>
CHR$(13) THEN NEXT I
1160 LINE (250,195)-(5,180),4,B
F:PRINT#1,"NOVA VELOC? (0 TERMI
NA):":SP$=""
1162 PRESET (210,180):PRINT#1,S
P$
1165 A$=INKEY$:IF A$=CHR$(13) T
HEN 1190
1170 IF A$<"0" OR A$>"9" THEN 1
165
1180 SP$=SP$+A$:GOTO 1162
1190 SP=VAL(SP$):IF SP=0 THEN R
=160
1195 IF SP<0 OR SP>999 THEN 116
0
1200 NEXT R
1210 RETURN
```

```
95  FOR I = 770 TO 795: READ A:
 POKE I,A: NEXT
100  TEXT : HOME
110  PRINT  TAB( 16);"M E N U"
120  VTAB 7: HTAB 5: PRINT "1 -
 MOVIMENTO HORIZONTAL PURO"
130  VTAB 9: HTAB 5: PRINT "2 -
 MOVIMENTO VERTICAL PURO"
140  VTAB 11: HTAB 5: PRINT "3
- MOVIMENTO COMBINADO"
150  VTAB 13: HTAB 5: PRINT "4
- ANGULOS VARIAVEIS"
160  GET A$: IF A$ < "1" OR A$
> "4" THEN 160
170  ON  VAL (A$) GOSUB 1000,20
00,3000,4000
180  GET A$: IF A$ =  CHR$ (13)
 THEN  RUN
190  GOTO 180
1000  HGR : HCOLOR= 3
1010  HPLOT 0,0 TO 279,0 TO 279
,159 TO 0,159 TO 0,0
1020 SP = 30
1030  HOME
1040  FOR R = 30 TO 150 STEP 24
:T =  - 1
1050  VTAB 21: PRINT "VELOC HOR
IZ: ";SP;" M/S": VTAB 24: PRINT
 "TECLE <CR> PARA DISPARAR ";
1060  GET A$: IF A$ <  > CHR$
(13) THEN 1060
1090 T = T + 1:X = SP * T
1100  HPLOT X / 5,R:W =  PEEK (
 - 16336):W =  PEEK ( - 16336)
1105  POKE  - 16368,0
1110  FOR D = 1 TO 20: IF  PEEK
 ( - 16384) < 128 THEN  POKE  -
 16368,0: NEXT
```



```
1120  IF SP * (T + 1) / 5 < 279
 THEN 1090
1130  IF R > 149 THEN 1200
1135  POKE  - 16368,0
1140  FOR D = 1 TO 300: IF  PEE
K ( - 16384) > 127 THEN D = 300

1150  POKE  - 16368,0: NEXT
1160  HTAB 1: VTAB 23: CALL  -
958: VTAB 23: INPUT "NOVA VELOC
IDADE (0 TERMINA) ";SP
1170  IF SP < 0 OR SP > 999 THE
N 1160
1190  IF SP = 0 THEN R = 160
1200  NEXT
1210  RETURN
5000  DATA  172,1,3,174,1,3,169
,4,32,168,252,173,48,192,232,20
8,253,136,208,239,206,0,3,208,2
31,96
```

Ao executar o programa, você verá um menu com quatro opções. Até agora, só a rotina para a primeira opção foi digitada. Assim, se teclar as opções 2, 3 ou 4, você obterá uma mensagem de erro. Ao pressionar 1, o programa vai para a sub-rotina 1000. Esta usa um laço **FOR...NEXT** (linhas 1040 a 1200) para permitir que um objeto seja lançado horizontalmente em seis velocidades diferentes. Na primeira vez, a velocidade de 30 m por segundo é usada automaticamente e simulada no vídeo como uma série de pontos em linha reta.

A seguir, você pode escolher uma velocidade diferente. Para sair do laço, pressione a tecla 0; o programa mostrará então o menu novamente. Digite um valor — 60, por exemplo — e pressione **<ENTER>** ou **<RETURN>** para disparar. A ação pode ser acelerada acionando-se continuamente a barra de espaços (tecla **<ENTER>** no Spectrum) ou teclando-a repetidas vezes. Agora, compare o resultado com a linha de velocidade 30 m/s. Escolha e compare cinco outras velocidades. Depois disso, o menu será reapresentado.

A parte do programa que desenha os pontos fica entre as linhas 1090 e 1120. A variável **T** simula o tempo que, neste caso, é incrementado de um em um segundo. Desse modo, os pontos são espaçados regularmente (o que é necessário para conservar a velocidade constante) na direção horizontal. A distância entre os pontos é determinada pela expressão **SP*T** na linha 1090. Ela faz com que, quanto maior a velocidade, maior o espaçamento entre os pontos.

A expressão **SP*T** pode ser reconhecida como parte da fórmula DISTÂNCIA = VELOCIDADE X TEMPO, como é ensinada nas aulas de física. Desse modo, fica bem claro que o programa usa o espaçamento entre os pontos, a fim de simular a velocidade.

MOVIMENTO VERTICAL

Digite as próximas linhas para simular o movimento em direção vertical:

```
2000 CLS
2020 LET G=10: LET SP=50
2030 PRINT '"ACELERACAO DA GRAV
.: m/s/s..."
2040 FOR R=3 TO 18 STEP 3
2050 PRINT AT 20,R;"*";AT 21,R;
G
2060 INPUT "<ENTER> PARA ATIRAR
", LINE I$
2080 FOR T=0 TO 250 STEP .5
2090 LET H=SP*T-0.5*G*T*T
2100 IF H>143 THEN  SOUND .05,1
0: PAUSE 50: NEXT T: GOTO 2110
2105 IF H>=0 THEN  PLOT R*8+4,H
+32: SOUND .05,H/4: PAUSE 40: N
EXT T
2110 IF R>15 THEN  GOTO 2180
2140 INPUT "NOVA GRAV. (0 SAI)"
, LINE I$
2142 IF LEN I$=0 THEN  GOTO 214
0
2145 LET G=VAL I$
2150 IF NOT (LEN I$>0 AND G>=0
AND G<400) THEN  GOTO 2140
2170 IF G=0 THEN  LET R=18
2180 NEXT R
2190 RETURN
```

```
2000 PCLS
2010 LINE(0,0)-(255,191),PSET,B
2020 G=10:SP=50:CLS
2030 PRINT @32,"ACELERACAO DA G
RAV.:":PRINT"M/S/S..."
2040 FOR R=0 TO 25 STEP 5
2050 PRINT @416+R,"*"+MID$(STR$
(G),2);
2060 PRINT @448,"<ENTER> PARA A
TIRAR"
2065 IF INKEY$<>CHR$(13) THEN 2
065
2070 SCREEN 1,0
2080 FOR T=0 TO 100 STEP .5
2090 H=SP*T-.5*G*T*T
2095 IF H<0 THEN T=250:GOTO 210
8
2100 IF H>159 THEN SOUND 250,1:
D=30 ELSE PSET (10+R*8,160-H,2)
:SOUNDH+10,1:D=50
2105 IF PEEK(345)=255 AND D>0 T
HEN D=D-1:GOTO 2105
2108 NEXT
2110 IF R>20 THEN 2180
2115 D=80
2119 A$=INKEY$
2120 IF INKEY$="" AND D>0 THEN
D=D-1:GOTO 2120
2140 PRINT @448,"":PRINT @448,"
NOVA GRAV.(0 SAI):";:INPUT G
2150 IF G<0 OR G>400 THEN 2140
2170 IF G=0 THEN R=26
2180 NEXT
2190 RETURN
```

```
2000 SCREEN 2,0,0
2010 LINE (5,0)-(250,160),15,B
2020 SP=50:G=10
2030 OPEN "GRP:" FOR OUTPUT AS
#1:PRESET (5,161):PRINT#1,"ACEL
 DA GRAVIDADE EM M/S/S"
2040 FOR R=0 TO 25 STEP 5
2050 PRESET (R*8,150):PRINT#1,G
:LINE (250,195)-(5,180),4,BF:PR
INT#1,"TECLE <RETURN> PARA DISP
ARAR"
2060 IF INKEY$<>CHR$(13) THEN 2
060
2080 FOR T=0 TO 100 STEP .5
2090 H=SP*T-.5*G*T*T
2095 IF H<0 THEN T=250:GOTO 210
8
2100 IF H>159 THEN PLAY"N90L19"
 ELSE PSET (30+R*8,160-H),15:A$
="N"+STR$(INT(H/2))+"L19":PLAYA
$
2105 FOR I=1 TO 100:IF INKEY$ <
>CHR$(32) THEN NEXT I
2108 NEXT T
2110 IF R>20 THEN 2180
2120 FOR I=1 TO 200:IF INKEY$ <
>CHR$(32) THEN NEXT I
2140 LINE (250,195)-(5,180),4,B
F:PRINT#1,"NOVA G? (0 TERMINA):
":G$=""
2142 PRESET (200,180):PRINT#1,G
$
2145 A$=INKEY$:IF A$=CHR$(13) T
HEN 2160LIST 2100-
2150 IF A$<"0" OR A$>"9" THEN 2
145
2155 G$=G$+A$:GOTO 2142
2160 G=VAL(G$):IF G=0 THEN R=26
2170 IF G<0 OR G>400 THEN 2140
2180 NEXT R
2190 RETURN
```

```
2000  HGR : HCOLOR= 3
2010  HPLOT 0,0 TO 279,0 TO 279
,159 TO 0,159 TO 0,0
2020 SP = 50:G = 10
2030  HOME : VTAB 21: PRINT "G:
 M/S/S"
2040  FOR R = 0 TO 25 STEP 5
2050  VTAB 21: HTAB R + 8: PRIN
T G: VTAB 24: PRINT "TECLE <CR>
 PARA DISPARAR ";
2070  GET A$: IF A$ <  >  CHR$
(13) THEN 2070
2080  FOR T = 0 TO 100 STEP .5
2090 H = SP * T - .5 * G * T *
T
2095  IF H < 0 THEN T = 250: GO
TO 2107
2100  IF H > 159 THEN  POKE 769
,200: POKE 768,2: CALL 770
2102  IF H < 160 THEN  HPLOT 50
 + R * 7,160 - H: POKE 769,H: P
OKE 768,2: CALL 770
2103  POKE  - 16368,0
2105  FOR D = 1 TO 20: IF  PEEK
 ( - 16384) < 128 THEN  POKE  -
 16368,0: NEXT
2107  NEXT T
```

```
2110  IF R > 20 THEN 2180
2115  POKE  - 16368,0
2120  FOR D = 1 TO 300: IF  PEE
K ( - 16384) > 127 THEN D = 300

2130  POKE  - 16368,0: NEXT
2140  HTAB 1: VTAB 23: CALL  -
958: VTAB 23: INPUT "NOVA ACELE
RACAO (0 TERMINA) ";G
2150  IF G < 0 OR G > 400 THEN
2140
2170  IF G = 0 THEN R = 26
2180  NEXT R
2190  RETURN
```

Execute o programa e escolha a opção 2 desta vez. Pressione <**ENTER**> ou <**RETURN**> para ver uma série de pontos traçados verticalmente na tela. Note que tais pontos não são espaçados regularmente, como na opção anterior; quanto mais alta sua localização, mais perto eles ficam uns dos outros.

Isso acontece porque o objeto é desacelerado pela ação da gravidade. E desta vez o som ajuda a explicar o que está acontecendo. À medida que o objeto sobe, a tonalidade do som torna-se mais aguda e, à proporção que ele desce, o som fica mais grave.

Como antes, a rotina lhe oferece seis tentativas (determinadas na linha 2040) de experimentar diferentes valores para o efeito da gravidade. Elas são armazenadas na variável **G**, que inicialmente é ajustada para 10 (linha 2020) e usada na linha 2090.

A relação é a fórmula para distância de um objeto em queda livre. Sua forma usual é:

$$S = VT + \frac{1}{2} GT^2,$$

onde S é a distância; V, a velocidade inicial; T, o tempo, e G, a aceleração da gravidade. TxT é usado na linha 2090 em vez de T↑2 ou T2, porque os computadores são mais ágeis para multiplicar do que para calcular potências. A aceleração de um objeto em movimento é calculada em função da mudança de velocidade e do tempo decorrido. Perto da superfície da Terra, g tem um valor aproximado 10 m/s/s. Isso significa que a velocidade de um objeto em queda aumenta de 10 m/s a cada segundo. E a velocidade de um objeto em ascensão diminui de 10 m/s a cada segundo. Esta é, portanto, uma aceleração negativa; por isso, o sinal + (mais) da fórmula padrão é substituída pelo − (menos) na linha 2090.

O cálculo de **G** é comumente usado em relação a viagens espaciais. A aceleração de uma espaçonave saindo da órbita terrestre atinge cerca de 10 g. Isso significa que essa aceleração é de 10x10 m/s/s ou 100 m/s/s (a notação m/s também é usada com freqüência).

Na primeira passada do laço que começa na linha 2040, a posição do objeto é traçada a intervalos de um segundo. A velocidade inicial é de 50 m/s e a aceleração é a da gravidade (10 m/s/s). Ambos os valores foram determinados na linha 2020. Como na primeira rotina, apressa-se a ação pressionando-se a barra de espaços (ou <**ENTER**>, no Spectrum). O valor de **G** pode ser mudado quando um novo valor é solicitado na tela. (Compare o efeito dos seis valores diferentes.) Antes de se completar o laço, pode-se voltar ao menu pressionando-se 0 para aceleração.

MOVIMENTO COMBINADO

Para simular o movimento de um projétil, é necessário apenas combinar essas rotinas de maneira que o objeto se mova na horizontal e na vertical, ao mesmo tempo. Digite estas linhas para montar a terceira rotina:

```
3000 CLS
3020 LET G=10: LET SP=50
3030 FOR R=1 TO 6
3040 PRINT AT 0,0;"GRAV.=";G;"m
/s/s "'"VELOCIDADE=";SP;"m/s "
3050 INPUT "<ENTER> PARA ATIRAR
", LINE I$
3070 FOR T=0 TO 250 STEP .5
3080 LET H=SP*SIN ((PI/180)*45)
*T-.5*G*T*T
3090 LET X=SP*COS ((PI/180)*45)
*T
3100 IF H<175 AND X<255 THEN  G
OTO 3105
3102 IF H>0 THEN  SOUND .05,10:
 PAUSE 25: NEXT T: GOTO 3110
3103 LET T=250: NEXT T: GOTO 31
10
3105 IF H>=0 THEN  PLOT X,H: SO
UND .05,H/4: PAUSE 40: NEXT T:
GOTO 3110
3106 LET T=250: NEXT T
3110 IF R=6 THEN  GOTO 3230
3120 PAUSE 50
3140 INPUT "NOVA GRAV. (0 SAI)"
, LINE I$
3150 IF LEN I$=0 THEN  GOTO 314
0
3155 LET G=VAL I$
3160 IF NOT (LEN I$>0 AND G>=0
AND G<1000) THEN  GOTO 3140
3170 IF G=0 THEN  LET R=6: GOTO
 3230
3190 INPUT "NOVA VELOC.", LINE
I$
3195 LET SP=VAL I$
3200 IF NOT (LEN I$>0 AND SP>0
AND SP<1000) THEN  GOTO 3190
3230 NEXT R
3240 RETURN
```

```
3000 PCLS
3010 LINE(0,0)-(255,191),PSET,B
3020 G=10:SP=50
3030 FOR R=1 TO 6:CLS
3040 PRINT "G    =";G;"M/S/S":P
RINT"VELOCIDADE=";SP;"M/S"
3050 PRINT @448,"<ENTER> PARA A
TIRAR"
3060 IF INKEY$<>CHR$(13) THEN 3
060
3065 SCREEN 1,0
3070 FOR T=0 TO 200 STEP .5
3080 H=SP*SIN(ATN(1))*T-.5*G*T*
T
3090 X=SP*COS(ATN(1))*T
3095 IF H<0 THEN T=250:GOTO 310
6
3100 IF X>251 THEN T=250:GOTO 3
106 ELSE IF H>189 THEN SOUND 25
0,1:D=25 ELSE PSET(X+2,190-H,(R
+3)/2):SOUND H+10,1:D=35
3104 IF PEEK(345)=255 AND D>0 T
HEN D=D-1:GOTO 3104
3106 NEXT
3110 IF R>5 THEN 3230
3115 D=100
3119 A$=INKEY$
3120 IF INKEY$="" AND D>0 THEN
D=D-1:GOTO 3120
3130 PRINT @416,"":PRINT @448,"
"
3140 PRINT @416,"NOVA GRAV. (0
SAI)";:INPUT G
3160 IF G<0 OR G>9999 THEN 3130
3170 IF G=0 THEN R=6:GOTO 3230
3180 PRINT @448,""
3190 PRINT @448,"NOVA VELOC.:";
:INPUT SP
3200 IF SP<0 OR SP>999 THEN 318
0
3230 NEXT
3240 RETURN
```

```
3000 SCREEN 2,0,0
3010 LINE (5,0)-(250,160),15,B
3020 SP=50:G=10
3025 OPEN "GRP:" FOR OUTPUT AS
#1
3030 FOR R=1 TO 6
3040 LINE (250,195)-(5,161),4,B
F:PRINT#1,"G =";G;"M/S/S     VE
L =";SP;"M/S"
3050 LINE (250,195)-(5,180),4,B
F:PRINT#1,"TECLE <RETURN> PARA
DISPARAR"
3060 IF INKEY$<>CHR$(13) THEN 3
060
3070 FOR T=0 TO 200 STEP .5
3080 H=SP*SIN(ATN(1))*T-.5*G*T*
T
3090 X=SP*COS(ATN(1))*T
3095 IF H<0 THEN T=250:GOTO 310
6
3100 IF X>245 THEN T=250:GOTO 3
106 ELSE IF H>159 THEN PLAY"N90
L19" ELSE PSET (X+6,160-H),15:A
$="N"+STR$(INT(H/2))+"L19":PLAY
```

```
A$
3104 FOR I=1 TO 100:IF INKEY$ <
>CHR$(32) THEN NEXT I
3106 NEXT T
3110 IF R>5 THEN 3230
3120 FOR I=1 TO 200:IF INKEY$ <
>CHR$(32) THEN NEXT I
3140 LINE (250,195)-(5,180),4,B
F:PRINT#1,"NOVA G? (0 TERMINA):
":G$="";
3142 PRESET (200,180):PRINT#1,G
$
3145 A$=INKEY$:IF A$=CHR$(13) T
HEN 3160
3150 IF A$<"0" OR A$>"9" THEN 3
145
3155 G$=G$+A$:GOTO 3142
3160 G=VAL(G$):IF G=0 THEN R=6:
GOTO 3230
3170 IF G<0 OR G>9999 THEN 3140
3180 LINE (250,195)-(5,180),4,B
F:PRINT#1,"NOVA VELOC?":SP$=""
3182 PRESET (200,180):PRINT#1,S
P$
3185 A$=INKEY$:IF A$=CHR$(13) T
HEN 32104180 RETURN
3190 IF A$<"0" OR A$>"9" THEN 3
185
3195 SP$=SP$+A$:GOTO 3182
3210 SP=VAL(SP$)
3220 IF SP<0 OR SP>999 THEN 318
0
3230 NEXT R
3240 RETURN
```

```
3000  HGR : HCOLOR= 3
3010  HPLOT 0,0 TO 279,0 TO 279
,159 TO 0,159 TO 0,0
3020 G = 10:SP = 50
3030  FOR R = 1 TO 6: HOME
3040  VTAB 21: PRINT "G= ";G;"
M/S/S    ";"VELOCIDADE= ";SP;" M
/S": VTAB 24: PRINT "TECLE <CR>
 PARA DISPARAR ";
3060  GET A$: IF A$ <  >  CHR$
(13) THEN 3060
3070  FOR T = 0 TO 200 STEP .5
3080 H = SP *  SIN ( ATN (1)) *
 T - .5 * G * T * T
3090 X = SP *  COS ( ATN (1)) *
 T
3095  IF H < 0 THEN T = 250: GO
TO 3125
3100  IF X > 275 THEN T = 250:
GOTO 3125
3105  IF H > 160 THEN  POKE 769
,200: POKE 768,2: CALL 770
3110  IF H < 160 THEN  HPLOT X
+ 2,160 - H: POKE 769,H: POKE 7
68,2: CALL 770
3115  POKE  - 16368,0
3120  FOR D = 1 TO 20: IF  PEEK
 ( - 16384) < 128 THEN  POKE  -
 16368,0: NEXT
3125  NEXT T
3130  IF R > 5 THEN 3230
3135  POKE  - 16368,0
3140  FOR D = 1 TO 300: IF  PEE
K ( - 16384) > 127 THEN D = 300

3145  POKE  - 16368,0: NEXT
3150  HTAB 1: VTAB 23: CALL  -
958: VTAB 23: INPUT "NOVA ACELE
RACAO (0 TERMINA) ";G
3160  IF G < 0 OR G > 9999 THEN
 3150
3170  IF G = 0 THEN R = 6: GOTO
 3230
3180  HTAB 1: VTAB 23: CALL  -
958: VTAB 23: INPUT "NOVA VELOC
IDADE (0 TERMINA) ";SP
3200  IF SP < 0 OR SP > 999 THE
N 3180
3230  NEXT R
3240  RETURN
```

Execute o programa e escolha a opção 3. Após **<ENTER>** ou **<RETURN>** terem sido pressionados, serão traçados pontos segundo uma curva que começa no canto inferior esquerdo da tela e termina em algum lugar em direção ao canto inferior direito. Essa é a trajetória de um objeto lançado a uma velocidade inicial de 50 m/s sob gravidade G.

ESTRUTURA DA ROTINA

A estrutura da rotina é similar às anteriores. O cálculo e o traçado são feitos em um laço **FOR...NEXT** (linhas 3030 a 3230), que permite comparar seis trajetórias diferentes, cinco das quais especificadas pelo usuário. Um valor 0 provoca o retorno ao menu. Mas é mais provável que você queira digitar um novo conjunto de valores para comparar as trajetórias.

Coloque o valor 5 em **G** e mantenha **SP** em 50; pressione **<ENTER>**. Desta vez, o objeto irá mais alto e mais longe. Mantenha **G** em 5 e reduza **SP** para 25. Compare os resultados.

Continue experimentando: modifique tanto **G** quanto **SP** e preste atenção aos sons. Estes foram especialmente planejados para ajudá-lo a compreender o movimento do objeto na tela. Dessa maneira, sons cada vez mais agudos indicam um movimento ascendente. A queda, por sua vez, é marcada por sons cada vez mais graves.

COMO FUNCIONA

Lembre-se de que a trajetória do objeto é traçada como coordenadas **H** no eixo Y e coordenadas **X** no eixo X. Essas duas variáveis são calculadas nas linhas 3080 e 3090. A única diferença entre elas é que a coordenada **H** tem a velocidade (**SP**) multiplicada pelo seno do ângulo e a coordenada **X** tem **SP** multiplicada pelo co-seno do mesmo ângulo (45°). Isso explica por que, quando **G** é pequena e **SP** grande, a trajetória parece uma linha diagonal a 45°.

SENO, CO-SENO, TANGENTE

Essas funções trigonométricas são necessárias para calcular a fração do movimento que se aplica a cada uma das duas direções. Tais frações, por sua vez, são chamadas de componentes horizontal e vertical do movimento. Se este tem uma velocidade inicial de 50 m/s, por exemplo, as duas componentes serão menores que 50. Somadas, elas dão exatamente 50 m/s.

Como você deve estar lembrado, a componente vertical é **SP*SEN A** e a horizontal, **SP*COS A**, onde A é o ângulo de inclinação do lançamento.

VISUALIZE O MOVIMENTO

Para compreender como esses valores são obtidos, é conveniente dar uma olhada em um esquema. O desenho da página 768 mostra um projétil iniciando seu movimento com velocidade real **V** em um ângulo **A** com a horizontal. As linhas tracejadas, por sua vez, mostram as componentes da velocidade (**Vh** e **Vx**) nas duas direções.

O seno do ângulo **A** é **Vh/V**; temos assim uma relação que pode ser lida como **Vh = V*SEN A**. Da mesma forma, o co-seno de A é **Vx/V**, de onde inferimos a equação **Vx = V*COS A**.

Seguindo esse padrão, temos **SP*SEN 45** para a componente vertical da velocidade e **SP*COS 45** para a componente horizontal.

No programa, isso aparece de um modo um pouco diferente. À exceção do Spectrum, usa-se a função **ATN(1)**. Essa função (arco tangente) fornece o ângulo cuja tangente é 1 (nosso caso): o ângulo de 45°. O computador, porém, utiliza tal valor em radianos. Por isso, recorremos à função — que fornece o valor na unidade adequada — e não ao valor 45, diretamente. No Spectrum usa-se o valor **pi/180**, que equivale a 1 grau, multiplicado por 45.

COMO MUDAR O ÂNGULO

Neste ponto, você pode estar querendo saber por que o ângulo foi fixado em 45°. Na verdade, tanto o ângulo quanto a velocidade inicial podem ser alterados para mudar a trajetória do projétil. Mais freqüentemente, varia-se o ângulo, deixando a velocidade constante. É o que faremos agora:

MICRO DICAS

COMO É FEITA A SIMULAÇÃO DE UM MOVIMENTO

Todos os programas que simulam fenômenos físicos de natureza contínua, como o movimento de um corpo no espaço, utilizam o mesmo tipo de modelo matemático: as equações de diferenças finitas. Por esse motivo, é interessante saber qual o seu significado e como funcionam. Assim, seus esforços de programação em muitas áreas, inclusive jogos, serão bem mais profissionais.

Uma equação de diferenças finitas diz qual é a lei matemática que rege a maneira pela qual uma variável muda com o tempo. Tomemos como exemplo o movimento de um corpo com velocidade constante. Suponhamos nesse caso que a distância percorrida pelo corpo em um intervalo de tempo bem pequeno — que chamaremos de **DT** — é **DD**. Como conseqüência, a equação que rege **DD** em função de **DT** é:

DD = V*DT,

onde V é a velocidade do corpo.

Na realidade, o que acontece na natureza é um movimento contínuo, ou seja, podemos imaginar um **DT** infinitesimalmente pequeno. Porém, se dermos agora o nome **D** à distância total percorrida pelo corpo, teremos a seguinte equação, que é bem mais útil que a primeira:

D = D + V*T

Essa é a forma a ser colocada em um programa de computador e se chama equação de diferenças finitas. Ela é calculada repetidamente, começando com o valor de **D** = 0. O valor de **D** é então incrementado a cada intervalo **DT** por um "pedacinho" calculado pela fórmula **V*DT**. Esse processo simples corresponde a uma integração da distância ao longo do tempo (integração é um termo específico da parte da matemática chamada cálculo diferencial e integral).

Como não é possível trabalhar em um computador com uma grandeza infinitamente pequena para **DT**, usa-se um **DT** pequeno mas finito.

```
4000 CLS
4010 LET FL=0
4020 RESTORE : FOR N=0 TO 7: RE
AD A: POKE USR "A"+N,A: NEXT N
4040 LET A=70: LET SP=50
4060 PRINT AT 0,0;"ANGULO=";A;C
HR$ 144;CHR$ 32
4070 INPUT "<ENTER> PARA ATIRAR
", LINE I$
4080 FOR T=0 TO 250 STEP .5
4090 LET H=SP*SIN ((PI/180)*A)*
T-.5*10*T*T: LET X=50*COS ((PI/
180)*A)*T
4100 IF H>=0 THEN  PLOT X,H+16:
 SOUND .05,H/4: PAUSE 40: NEXT
T: GOTO 4110
4105 LET T=250: NEXT T
4110 PAUSE 50
4130 INPUT "NOVO ANGULO (0 SAI)
", LINE I$
4135 IF LEN I$=0 THEN  GOTO 413
0
4140 LET A=VAL I$
4150 IF NOT (LEN I$>0 AND A>=0
AND A<90) THEN  GOTO 4130
4160 IF A=0 THEN  LET FL=1
4170 IF NOT FL THEN  GOTO 4060
4180 RETURN
5000 DATA 24,36,36,36,24,0,0,0
```

```
4000 PCLS
4020 LINE (0,0)-(255,191),PSET,
B
4040 A=70:SP=50
4060 CLS:PRINT "ANGULO = ";A;"G
RAUS"
4070 PRINT @448,"<ENTER> PARA A
TIRAR"
4072 IF INKEY$<>CHR$(13) THEN 4
072
4075 SCREEN 1,0:AN=A*ATN(1)/45
4080 FOR T=0 TO 250 STEP .5
4090 H=SP*SIN(AN)*T-.5*10*T*T:X
=SP*COS(AN)*T
4092 IF H<0 THEN T=250:GOTO 410
0
4094 IF X>251 THEN T=250:GOTO 4
100 ELSE IF H>189 THEN SOUND 25
0,1:D=25 ELSE PSET(X+2,190-H,2)
:SOUND H+10,1:D=35
4096 IF PEEK(345)=255 AND D>0 T
HEN D=D-1:GOTO 4096
4100 NEXT
4110 A$=INKEY$
4120 IF INKEY$="" THEN 4120
4130 PRINT @448,"NOVO ANGULO (0
 SAI):";:INPUT A
4160 IF A<0 OR A>=90 THEN 4120
4170 IF A>0 THEN 4060
4180 RETURN
```

```
4000 SCREEN 2,0,0
4020 LINE (5,0)-(250,160),15,B
4040 SP=50:A=70
4050 OPEN "GRP:" FOR OUTPUT AS
#1
4060 LINE (250,175)-(5,161),4,B
F:PRINT#1,"ANGULO =";A;"GRAUS"
4070 LINE (250,195)-(5,180),4,B
F:PRINT#1,"TECLE <RETURN> PARA
DISPARAR"
4072 IF INKEY$<>CHR$(13) THEN 4
072
4075 AN=A*ATN(1)/45
4080 FOR T=0 TO 250 STEP .5
4090 H=SP*SIN(AN)*T-.5*10*T*T:X
=SP*COS(AN)*T
4092 IF H<0 THEN T=250:GOTO 410
0
4094 IF X>245 THEN T=250:GOTO 4
100 ELSE IF H>159 THEN PLAY"N90
L19" ELSE PSET (X+6,160-H),15:A
$="N"+STR$(INT(H/2))+"L19":PLAY
A$
4096 FOR I=1 TO 100:IF INKEY$ <
>CHR$(32) THEN NEXT I
4100 NEXT T
4130 LINE (250,195)-(5,180),4,B
F:PRINT#1,"NOVO ANGULO? (0 TERM
INA):":G$=""
4132 PRESET (220,180):PRINT#1,G
$
4135 A$=INKEY$:IF A$=CHR$(13) T
HEN 4150
4140 IF A$<"0" OR A$>"9" THEN 4
135
4145 G$=G$+A$:GOTO 4132
4150 A=VAL(G$):IF A=0 THEN 4180
4160 IF A<0 OR A>89 THEN 4130
4170 GOTO 4060
4180 RETURN
```

```
4000  HGR : HCOLOR= 3
4020  HPLOT 0,0 TO 279,0 TO 279
,159 TO 0,159 TO 0,0
4040 SP = 50:A = 70
4060  HOME : VTAB 21: PRINT "AN
GULO =";A;" GRAUS  "
4065  VTAB 24: PRINT "TECLE <CR
> PARA DISPARAR ";
4070  GET A$: IF A$ <  >  CHR$
(13) THEN 4070
4075 AN = A *  ATN (1) / 45
4080  FOR T = 0 TO 250 STEP .5
4090 H = SP *  SIN (AN) * T - .
5 * 10 * T * T:X = SP *  COS (A
N) * T
4092  IF H < 0 THEN T = 250: GO
TO 4100
4093  IF X > 275 THEN T = 250:
GOTO 4100
4094  IF H > 160 THEN  POKE 769
,200: POKE 768,2: CALL 770
4095  IF H < 160 THEN  HPLOT X
+ 2,160 - H: POKE 769,H: POKE 7
68,2: CALL 770
4097  POKE  - 16368,0
4098  FOR D = 1 TO 20: IF  PEEK
 ( - 16384) < 128 THEN  POKE  -
 16368,0: NEXT
4100  NEXT T
4130  HTAB 1: VTAB 23: CALL  -
958: VTAB 23: INPUT "NOVO ANGUL
O (0 TERMINA) ";A
4160  IF A < 0 OR A > 89 THEN 4
130
4170  IF A > 0 THEN 4060
4180  RETURN
```

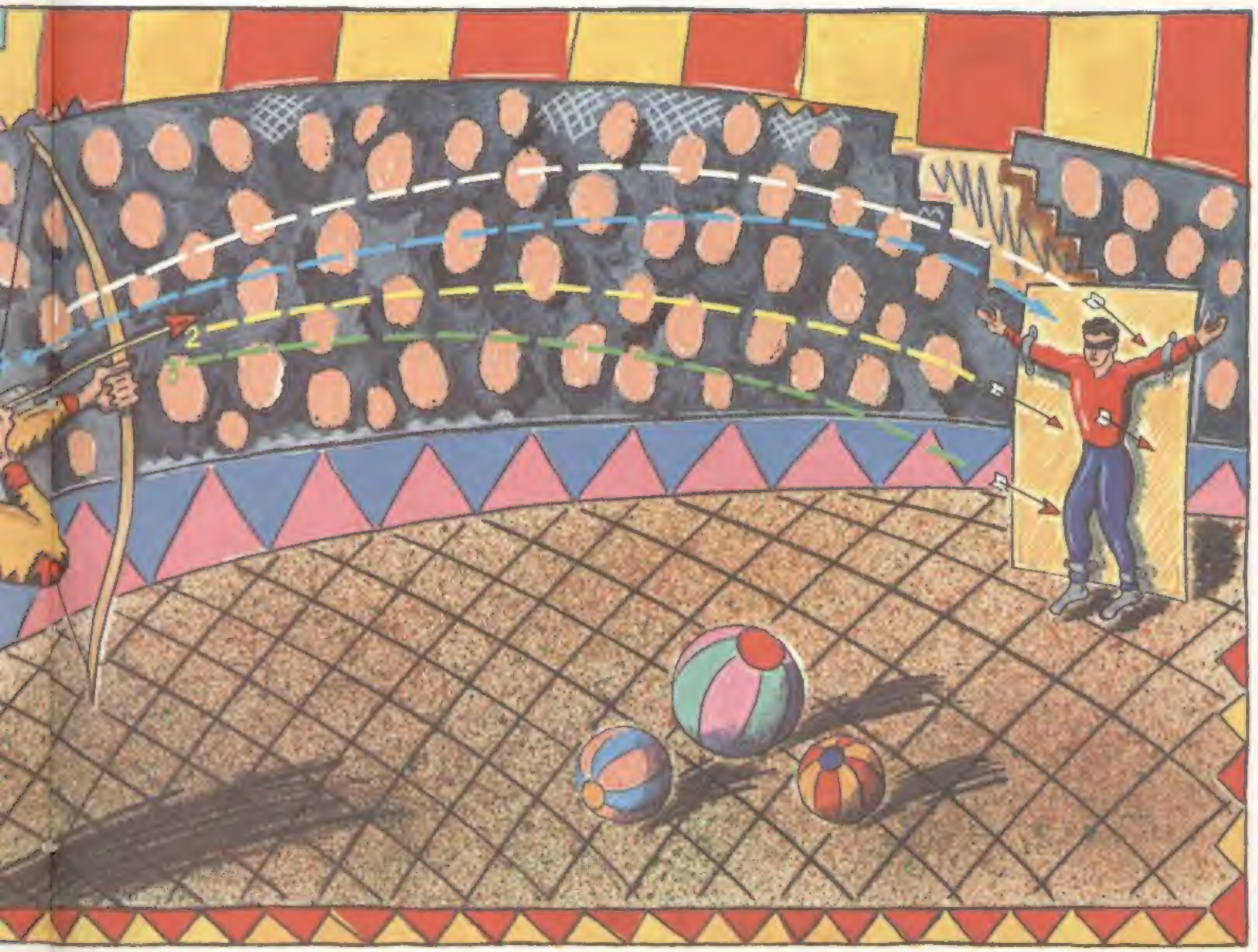
A MAIOR DISTÂNCIA

Ao executar a quarta opção, você poderá ver a trajetória de um objeto lançado com uma velocidade inicial de 50 m/s e a uma inclinação de 70° (ambos os valores determinados na linha 4040 e usados na linha 4090).

A rotina funciona como as precedentes, exceto pelo fato de que se pode dar entrada a tantos ângulos quantos se desejar (entre 1° e 89°). Assim, toda vez que essa rotina for usada, a variável A da linha 4090 armazenará o ângulo que você escolheu. Desta vez, a rotina é um laço infinito, de modo que só se volta ao menu teclando-se 0 para um ângulo.

Usando esta rotina, tente encontrar o ângulo que lhe dá a maior distância de percurso nas direções vertical e horizontal. Será fácil descobri-lo: é o nosso conhecido ângulo de 45°.

Esse ângulo, porém, permite obter a maior distância apenas no caso em que os pontos de partida e de chegada estão no mesmo nível, pois é preciso levar em conta a resistência do ar.



ACASO E PROBABILIDADE

Um dos pioneiros da matemática moderna, o francês Blaise Pascal foi o criador de um triângulo de números extremamente importante para o cálculo das probabilidades.

A força de um computador está na sua capacidade de obedecer instruções repetidamente, com precisão e velocidade. Entretanto, quando comparado aos processos instantâneos de raciocínio do ser humano, o computador pode parecer lento e obsoleto. A mente humana é boa principalmente no que diz respeito a julgar e comparar parâmetros, tais como distância, velocidade e intensidade de luz.

Mas até o mais sofisticado dos órgãos comete erros ao calcular o resultado de eventos. Ainda assim, é essencial, por exemplo, poder dizer que, "para fins de seguro de vida, uma pessoa vive de 65 a 70 anos", ou que "não é provável que a terra volte a tremer no México em 1986". Tais afirmações são muito comuns no nosso dia-a-dia. Além disso, são importantes para fins sociais, comerciais e científicos. Assim, quando empregamos expressões como *prós e contras, estimativa, dúvida, expectativa*, estamos fazendo cálculos mentais de probabilidade.

A PROBABILIDADE EXPLICADA

Probabilidade é uma ferramenta científica para se medir o acaso. Usada para calcular o provável resultado de um evento, ela se baseia na existência de um número mensurável de resultados, como num jogo de futebol, ou num lançamento de dados, ou de moedas, num jogo de cartas etc. Naturalmente, temos que poder medir ou quantificar os resultados; por isso, eventos como corridas de cavalo ou jogos de futebol são assuntos difíceis para o cálculo de probabilidades. Quando dizemos "espero vencer", na verdade, estamos afirmando ser alta a *probabilidade* de conseguir a vitória e não apenas nosso desejo de triunfar.

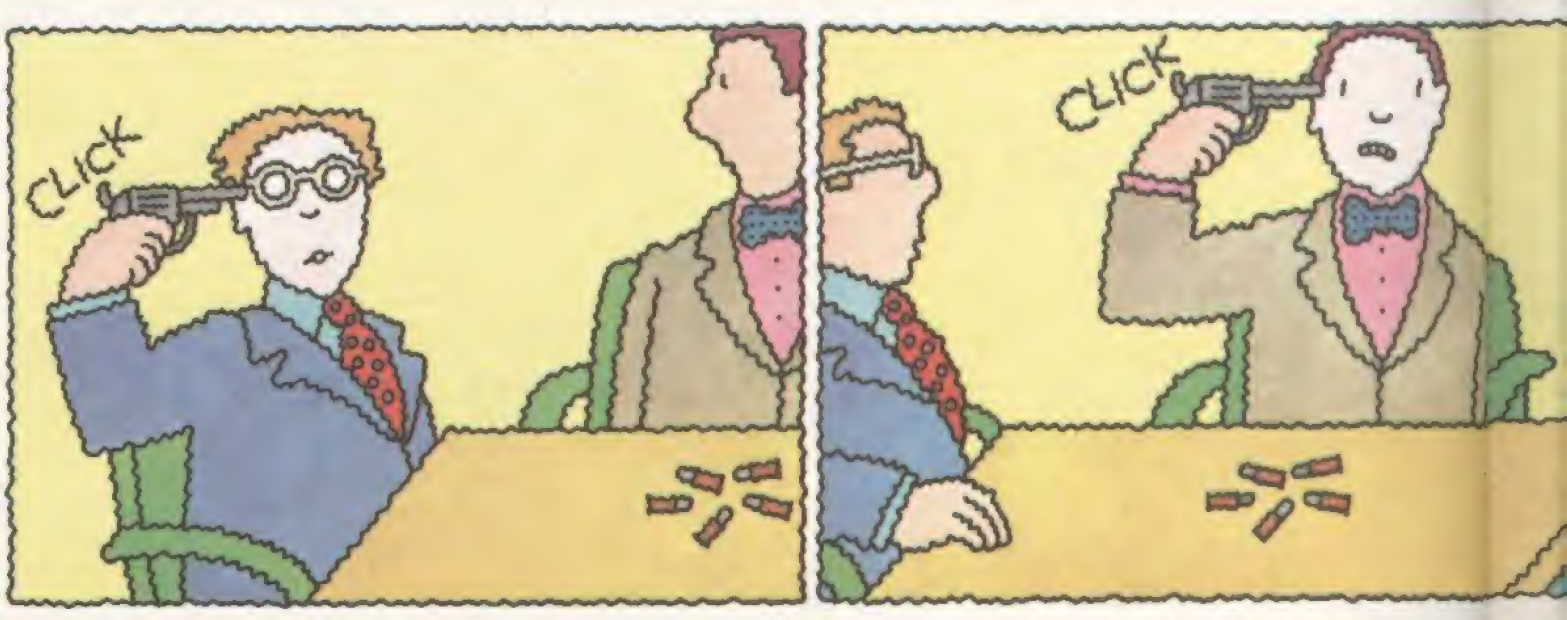
A maioria das pessoas apóia-se na intuição como principal ferramenta no cálculo do acaso (ou probabilidade). Entretanto, se quisermos examinar os efeitos possíveis e, dentre eles, aqueles que são mais prováveis, num problema de probabilidade, podemos chegar a um resultado bastante preciso. Não é difícil aprender a prever os resultados mais prováveis, e mesmo que não possamos garanti-los, a possibilidade de acerto é bem maior que a de uma estimativa feita a esmo.

PROBABILIDADE E COMPUTAÇÃO

Qual será então a relação entre probabilidade e computação? Embora o tipo de probabilidade descrito até agora seja muito vago e dependente de estimativas, é possível deduzir fórmulas matemáticas para determinados acontecimentos que nos permitam prever os resultados mais prováveis com um certo grau de precisão.

Existem duas maneiras pelas quais o computador pode ser útil aqui. A primeira delas consiste em programá-lo para simular o próprio evento — é bem mais cômodo fazer com que a máquina jogue um dado 2000 vezes e depois analise o resultado do que termos nós mesmos que fazê-lo. A segunda maneira, entretanto, só funciona quando conhecemos as fórmulas para um certo evento; nesse caso, podemos fazer com que o computador calcule o resultado.

Visto por alguns como um exercício puramente teórico, esse tipo de cálculo serve de base para muitas aplicações úteis do computador. Uma dessas aplicações consiste nos jogos onde, por exemplo, os pontos ganhos dependem da probabilidade de certos resultados. Do mesmo modo, poderíamos escrever um programa que estabelecesse as probabilidades de chover num determinado dia. Por enquanto, porém, vamos concentrar nossa atenção apenas na teoria. Mais adiante, trataremos da construção de alguns desses aplicativos.



- O QUE É PROBABILIDADE E COMO MEDI-LA
- PROBABILIDADE, FREQÜÊNCIA E ACASO
- PROGRAMA LANÇADOR DE MOEDAS
- PROBABILIDADE DE VÁRIOS RESULTADOS
- TRIÂNGULO DE PASCAL
- DISTRIBUIÇÃO DE FREQÜÊNCIA
- PREVEJA O RESULTADO

COMO MEDIR A PROBABILIDADE

A teoria de medição da probabilidade requer que sejam conhecidos os resultados possíveis de um evento, e que eles ocorram com uma certa freqüência, passível de ser medida. A probabilidade de acontecer um certo evento é o número de vezes com que ele se repete (freqüência), comparado com o total de todos os resultados possíveis. Ou, em outras palavras, é a freqüência expressa em fração do conjunto dos resultados possíveis.

Devemos notar que, se um acontecimento se repetir toda vez que ocorrerem certas circunstâncias, então sua probabilidade será igual a 1. Isso porque sua freqüência será igual ao número de resultados possíveis, e a divisão de dois números iguais resulta em 1. Temos, então, que 1 é a maior probabilidade possível, e que a soma das frações de probabilidade de todos os resultados possíveis resultará invariavelmente nesse número.

Um dos métodos mais simples e antigos de visualizar a probabilidade é lançar uma moeda e prever o resultado, ou seja, que face terá ao cair: cara ou coroa. Como a moeda tem só dois lados, podemos inferir que, qualquer que fosse o número de lançamentos, deveríamos obter cara, metade das vezes, e coroa a outra metade, ignorando a remota possibilidade de a moeda cair em pé. Para ilustrar melhor esse método, digite e rode o primeiro programa:

```
10 COLOR 1,3,3:CLS
20 INPUT"QUAL O TESTE (1-4) ";X
30 CLS:IF X<1 OR X>4 THEN 10
40 ON X GOTO 70,70,180,460
50 REM.........probabilidade
60 REM...lançamento de moedas
70 H=0:T=0
80 LOCATE5,3:PRINT"PRESSIONE A
TECLA DE ESPAÇO"
85 LOCATE9,4:PRINT"PARA LANÇAR
A MOEDA"
90 LOCATE13,7:PRINT"CARAS:  0":
PRINTTAB(13)"COROAS: 0"
100 A$=INKEY$:IFA$=CHR$(13) THE
N SCREEN0:RUN
105 IFA$<>" "THEN 100
110 IF X=2 THEN FORN=1 TO 100
120 IFINT(RND(1)*2)+1=1 THEN H=
H+1 ELSE T=T+1
130 LOCATE20,7:PRINTH:LOCATE20,
8:PRINTT
140 IF X=1 THEN 100
160 NEXTN:GOTO 100
```

```
5  HOME
20  INPUT "QUAL O TESTE (1-4) ?
 ";X
30  HOME : IF X < 1 OR X > 4 TH
EN 5
40  ON X GOTO 70,70,180,460
50  REM .........PROBABILIDADE
60  REM ..LANCAMENTO DE MOEDAS
70 H = 0:T = 0
80  PRINT "PRESSIONE A TECLA DE
 ESPACO"
85  PRINT "    PARA LANCAR A MO
EDA"
90  VTAB (4): PRINT "CARAS:  0"
: PRINT "COROAS: 0"
100  GET A$: IF A$ =  CHR$ (13)
 THEN  TEXT : RUN
105  IF A$ <  > " " THEN 100
110  IF X = 2 THEN  FOR N = 1 T
O 100
120  IF  INT ( RND (1) * 2) + 1
 = 1 THEN H = H + 1: GOTO 130
125 T = T + 1
130  VTAB (4): HTAB (9): PRINT
;H: HTAB (9): PRINT ;T
140  IF X = 1 THEN 100
160  NEXT N: GOTO 100
```

```
10 PMODE 3,1:CLS
20 INPUT"QUAL O TESTE (1-4) ";X
30 CLS:IF X<1 OR X>4 THEN 20
40 ON X GOTO 70,70,180,460
50 REM .... PROBABILIDADE
60 REM .... LANCAMENTO DA MOEDA
70 H=0:T=0
```

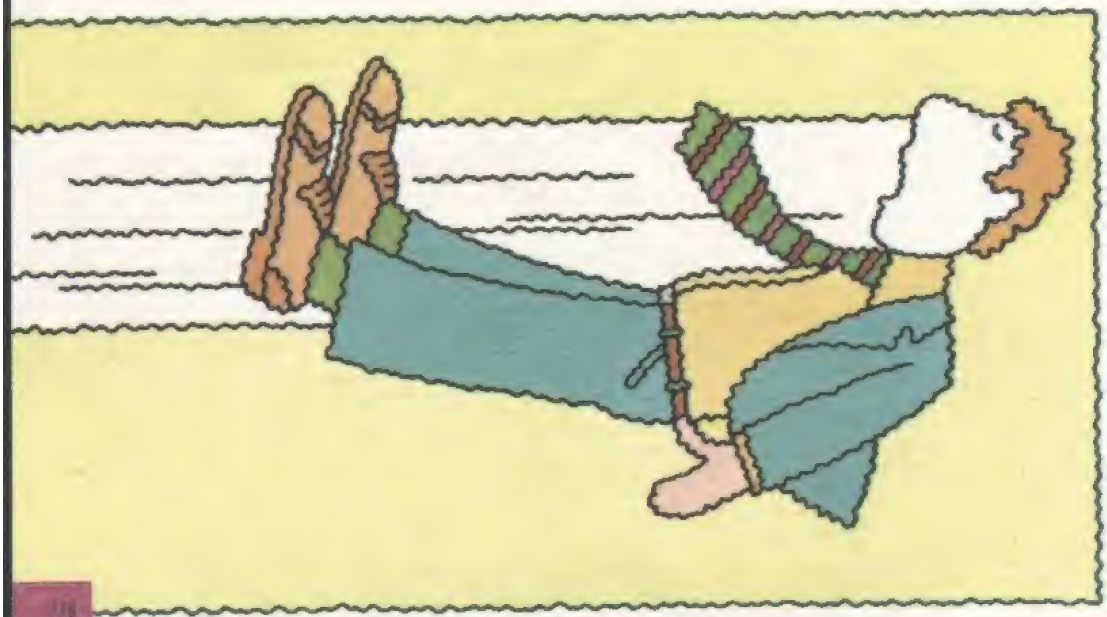
```
80 PRINT @65,"PRESSIONE A BARRA
 DE ESPACOS    PARA LANCAR A MO
EDA"
90 PRINT @288,"CARAS - 0":PRINT
"COROAS- 0"
100 A$=INKEY$:IF A$<>" " THEN 1
00
110 IF X=2 THEN FOR N=1 TO 100
120 IF RND(2)=1 THEN H=H+1:PRIN
T @204,"CA":PRINT @295,H;ELSE T
=T+1:PRINT @207,"CO":PRINT @327
,T;
130 IF X=1 AND PEEK(345)<>247 T
HEN 130
140 PRINT @204,""
150 IF X=1 THEN FOR D=0 TO 400:
NEXT:GOTO 120 ELSE NEXT
160 A$=INKEY$:IF A$<>CHR$(13) T
HEN 160 ELSE END
```

```
 5 BORDER 7: PAPER 7: INK 9:
 CLS
10 DIM n(4)
20 RESTORE 9000: FOR n=1 TO 4
: READ n(n): NEXT n
30 INPUT "Qual o teste (1 a 4
)?",x: CLS
40 BORDER x: GOTO n(x)
50 REM Probabilidade
60 REM Lancamento de Moeda
70 LET h=0: LET t=0
80 PRINT AT 2,1;"Pressione <S
PACE> para lancar"
90 PRINT AT 20,10;"CARAS - 0"
;AT 21,10;"COROAS- 0"
100 IF INKEY$<>CHR$ 32 THEN
GOTO 100
110 IF x=2 THEN  FOR n=1 TO
 100
120 IF INT (RND*2)=1 THEN  LET
h=h+1: PRINT AT 10,15;"CA";AT
20,18;h: GOTO 130
125 LET t=t+1: PRINT AT 12,15;
"CO";AT 21,18;t
130 IF x=1 THEN  IF INKEY$<>
CHR$ 32 THEN  GOTO 130
140 PRINT AT 10,15;"  ";AT 12,
15;"  "
150 IF x=1 THEN  FOR m=1 TO
 100: NEXT m: GOTO 120
155 NEXT n: STOP
9000 DATA 70,70,170,460
```

Este programa será incrementado mais adiante. Ao rodá-lo, seremos perguntados a respeito do número do teste que desejamos fazer. Esse primeiro programa contém somente os dois testes iniciais. Digite 1 e estaremos prontos para lançar a moeda — usando a tecla de espaço. O ponto crucial do programa está na linha 120, que gera aleatoriamente valores 1 (para cara) e 0 (para coroa), além de fazer a contagem do número de caras e coroas obtidas). A linha 150 no programa do Spectrum e também no do TRS-Color provoca uma pausa entre as jogadas.

Poucos lançamentos, provavelmente, fornecerão valores diferentes para cara e coroa, representados no programa pelas variáveis **H** e **T**, respectivamente. Quanto maior o número de jogadas, mais próximas uma da outra serão as freqüências de **H** e **T** — tendendo, cada uma, à metade do número total de lançamentos (ou 50%). Para demonstrar esse efeito, rode o programa novamente, mas, desta vez, selecione o teste 2. Ao ser pressionada a tecla de espaço, um laço preparado pela linha 110 lançará a moeda cem vezes. Note que as duas variáveis, **H** e **T**, estão bem próximas de 50. Mude o 100 da linha 110 para 1000, rode o programa e observe como cara e coroa tendem para 500.

Pode acontecer que, num teste de poucas jogadas, todas elas sejam coroa. Apesar disso, a probabilidade de se obter coroa será sempre igual a 1/2 (meio). Convém não nos esquecermos disso, principalmente quando estivermos trabalhando com mais de um evento. Muitos acreditam que, se dez lançamentos seguidos de uma moeda derem cara, a possibilidade de se obter coroa na décima-primeira jogada será maior do que antes. Isso não é verdade; a possibilidade de dar coroa ainda será de 50%. Os eventos passados não têm influência sobre os lançamentos seguintes. No entanto, se lançarmos onze moedas ao mesmo tempo, a chance de se obter onze coroas será menor que a de obter dez coroas e uma cara. A maior chance é de que o número de coroas seja próximo ao número de caras.



MÚLTIPLOS EVENTOS

Quando há vários eventos, algumas informações adicionais são necessárias para se poder prever a probabilidade de cada resultado. Uma informação essencial é o número total de resultados possíveis. Por exemplo, se lançarmos duas vezes uma moeda, teremos três resultados possíveis: duas caras, uma cara e uma coroa, e duas coroas. À primeira vista, pareceria que cada resultado pode ocorrer 1/3 das vezes. Na verdade, as probabilidades são duas caras acontecerem 1/4 (25%) das vezes; duas coroas, 1/4; e uma cara e uma coroa, 1/2. Para entender essa terceira probabilidade, precisamos de outra informação essencial, que dê conta de quantas vezes cada resultado ocorre. Uma cara e uma coroa acontecem duas vezes, porque existem duas maneiras de se obter tal resultado: uma cara e depois uma coroa, ou uma coroa e depois uma cara. Conseguimos assim um total de quatro resultados, três dos quais diferentes entre si.

Na prática, costuma-se usar dois recursos matemáticos que nos poupam o trabalho de calcular a ocorrência de cada resultado: o teorema binomial e o triângulo de Pascal. Binomial significa "dois termos". Se um evento tem só dois resultados possíveis e conhecemos a probabilidade de cada um deles, podemos usar o teorema binomial para calcular tais probabilidades. O teorema binomial nos diz o que devemos esperar de testes repetidos de um evento com dois resultados possíveis. Chamaremos **P** à probabilidade de um dos resultados e **Q** à do outro (lembremos que a soma de **P** e **Q** deve ser igual a 1). Denominaremos **N** o número de eventos.

No exemplo do lançamento de uma moeda, **P** (digamos, a possibilidade de dar cara) e **Q** (coroa) serão iguais a 1/2 para cada jogada. De acordo com o teo-

rema binomial, a chance de que qualquer evento aconteça duas vezes seguidas é a probabilidade de que ocorra uma vez multiplicada por ela mesma. Em geral, a regra consiste na probabilidade do evento elevado à potência **N**. Portanto, para o caso de duas caras seguidas, **P↑N = 1/2 x 1/2 = 1/4**. Existe então uma chance em quatro de que obtenhamos duas caras seguidas. Do mesmo modo, a possibilidade de se conseguir cinco coroas sucessivas é **P↑N**=(1/2 ↑ 5), ou 1/32.

Como veremos mais tarde, podemos usar esse método para calcular a probabilidade em qualquer caso onde existam somente dois resultados possíveis (sim/não ou cara/coroa, por exemplo). Mas qual seria a chance de se obterem três caras e duas coroas em cinco lançamentos? Para responder a essa pergunta, precisamos de um modelo mais complexo. Muito útil neste caso é o triângulo de números, desenvolvido pelo matemático, físico e filósofo francês Blaise Pascal. Esse triângulo fornece todos os resultados possíveis de um evento binomial, e pode ser entendido como algumas fileiras de números. As primeiras sete fileiras são:

```
Fileira 0)                1
Fileira 1)              1   1
Fileira 2)            1   2   1
Fileira 3)          1   3   3   1
Fileira 4)        1   4   6   4   1
Fileira 5)      1   5  10  10   5   1
Fileira 6)    1   6  15  20  15   6   1
```

Para construir um triângulo como esse, escreva primeiramente as fileiras 0 e 1, as mais fáceis de serem lembradas. A fileira 2 começa com um 1 à esquerda da fileira 1 e termina com um 1 à direita. O número do meio é obtido pela soma dos dois algarismos imediatamente acima dele (1 + 1). Da mesma maneira, a fileira 6 é obtida somando-se 5 + 1, 5 + 10, 10 + 10, 10 + 5 e 5 + 1. Se continuarmos com esse processo, teremos um triângulo com um número de fileiras cada vez maior, o que seria difícil de conseguir com a utilização de qualquer outro método.

O triângulo de Pascal nos dá todas as informações de que precisamos quando lançamos diversas moedas (ou uma moeda várias vezes). O número de moedas (ou de lançamentos) estabelece o número da fileira que devemos olhar; o número de itens na fileira determina o de resultados diferentes. Por exemplo, existem duas possibilidades para uma moeda (1 e 1 na fileira 1) e sete para seis moedas (1, 6, 15, 20, 15, 6, 1 na fileira 6). A soma dos itens na fileira nos dá o número total de resultados (2 para uma moeda, 4 para duas moedas, e assim por diante).

Cada número na fileira define a probabilidade. Por exemplo, na fileira 2, o primeiro número (1) é a probabilidade para duas caras; o segundo (2), a probabilidade para uma cara e uma coroa; o terceiro (1), para duas coroas. Evidentemente, a freqüência tem que ser dividida pelo número total de resultados (4, neste caso) para se obter a probabilidade. Note que o resultado da adição dos números em cada fileira é sempre uma potência de dois (1, 2, 4, 8, 16). Isso acontece porque, para qualquer um dos eventos, há só duas opções: cara ou coroa.

A praticidade desse método é posta à prova quando queremos, por exemplo, calcular as probabilidades de lançamento de trinta moedas; a construção de um triângulo com trinta fileiras seria muito trabalhosa, além de tomar muito espaço. Existe, porém, um método gráfico ao qual podemos recorrer em tais situações, e é aqui que entra a ajuda do computador.

CURVAS DE DISTRIBUIÇÃO

Onde existirem muitos resultados com probabilidades não muito claras, conseguiremos quase sempre calcular o que precisamos, plotando (ou seja, marcando num gráfico) uma curva de distribuição, definida pela freqüência dos resultados que conhecemos. Como em qualquer método gráfico, as informações nele contidas são percebidas logo de início. Se, por exemplo, uma moeda for lançada trinta vezes (o que seria o mesmo que lançar trinta moedas de uma só vez), podemos traçar um gráfico com o número de caras obtidas. Digite a pró-

xima seção de programa, mas não apague a anterior:

```
170 REM......picos randômicos
180 SCREEN2:FORX=5 TO 255 STEP
4
190 GM=0:GOSUB 610
200 FORY=0 TO H*6 STEP 2
210 PSET(X,188-Y),1
220 NEXT Y,X
230 IFINKEY$=""THEN230 ELSE RUN
610 H=0:T=0
650 FORS=1 TO 30
660 IFINT(RND(1)*2)+1=1 THEN H=
H+1 ELSE T=T+1
670 NEXT
680 RETURN
```

```
170  REM .....PICOS ALEATORIOS
180  HGR : HCOLOR= 3: FOR X = 4
 TO 275 STEP 10
190 GM = 0: GOSUB 610
200  FOR N = 0 TO H: HPLOT X,16
0 - (N * 6): NEXT N
220  NEXT X
230  GET A$: IF A$ = "" THEN 23
0
235  TEXT : RUN
610  REM .....LANCAMENTO
620 H = 0:T = 0
630  HOME : VTAB (22): PRINT "C
ARAS:   ": PRINT "COROAS: "
640  IF GM <  > 0 THEN  VTAB (2
2): HTAB (31): PRINT "JOGO: ";G
M: HTAB (17): PRINT "CARAS EM 3
0 LANCAMENTOS"
650  FOR S = 1 TO 30
660  IF  INT ( RND (1) * 2) + 1
 = 1 THEN H = H + 1: GOTO 668
665 T = T + 1
668  VTAB (22): HTAB (10): PRIN
T ;H: HTAB (10): PRINT ;T
670  NEXT S
680  RETURN
```

```
170 REM .... GRAFICO
180 PCLS4:SCREEN 1,0:FOR X=0 TO
 255 STEP 4
190 GM=0:GOSUB 610
200 FOR Y=0 TO H*6 STEP 2
210 PSET(X,188-Y,2)
220 NEXT Y,X
230 GOTO 160
610 H=0:T=0
650 FOR TS=1 TO 30
660 IF RND(2)-1 THEN H=H+1 ELSE
 T=T+1
670 NEXT
680 RETURN
```

```
170 REM Grafico
175 PLOT 0,0: DRAW 180,0
180 FOR x=4 TO 160 STEP 4
190 LET gm=0: GOSUB 610
200 FOR n=0 TO h: PLOT x,n*6:
NEXT n
220 NEXT x
230 STOP
610 REM Lancamento
620 LET h=0: LET t=0
630 PRINT AT 4,22;"CARAS -";h;
AT 6,22;"COROAS-";t
640 IF gm<>0 THEN  PRINT AT 0,
0;"JOGADAS-";gm;AT 21,3;"CARAS
EM 30 LANCAMENTOS:"
650 FOR s=1 TO 30
660 IF RND>=.5 THEN  LET h=h+1
: PRINT AT 2,24;"  CA";AT 4,29
;h;" ": GOTO 670
665 LET t=t+1: PRINT AT 2,24;"
CO  ";AT 6,29;t;" "
670 NEXT s
680 RETURN
```

Ao rodar o programa, desta vez, selecione o teste 3. Devemos ver na tela um gráfico com uma série de pontos chegando a vários picos. Esta é uma das muitas formas possíveis nesse tipo de análise. Os picos são os números de caras em cada trinta lançamentos ao longo do eixo Y, espaçados igualmente ao longo do eixo X. Existem mais picos altos do que baixos. A razão disso é que a possibilidade de se obter cerca de quinze (ou de doze a dezessete) caras é muito maior que a de se obter um número menor ou maior. Isso também pode ser observado no triângulo de Pascal, onde os valores mais altos se encontram na região central.

A linha 180 gera um laço para espaçar os pontos ao longo do eixo X. A variável **GM** (número de jogadas) é zerada na linha 190 e uma rotina (linhas 610 a 680) é chamada para executar cada bateria de trinta lançamentos. Essa rotina usa os elementos do segundo teste, mas lança a moeda "eletrônica" trinta vezes e não cem. Com exceção do TRS-COLOR e do MSX, ao rodarmos esse teste, notaremos o placar para cara e coroa que aparecerá na tela. Uma vez atingidos os trinta lançamentos, o número de caras que foi acumulado na rotina sofrerá um ajuste de escala na linha 200 e será plotado na coordenada Y pela linha 210 (linha 200, no Spectrum e no Apple).

Para tirar o máximo proveito de uma análise desse tipo, precisamos rearranjar as informações, de modo que obtenhamos uma das curvas mais conhecidas no meio estatístico: a distribuição normal. Digite mais estas linhas para obtermos tal curva:

```
450 REM...distribuição normal
460 DIM G(30)
470 CLS:SCREEN2:LINE(6,0)-(6,19
1),15:LINE-(255,191),15
480 GOSUB 560
485 A$=INKEY$:IFA$<>" "THEN 485
560 DEF FNN(X)=1/(4.4429*2.718^
((X*X)/2))
570 DRAW"BM7,190":FORX=2 TO 255
 STEP2
580 LINE-(X,191-640*FNN((X-127)
/24)),1
590 NEXT:RETURN
```

```
450  REM ..DISTRIB NORMAL
460  DIM G(30): HGR : HCOLOR= 3

470  HPLOT 0,0 TO 0,159: HPLOT
 TO 279,159
480  GOSUB 560
485  GET A$: IF A$ <  > " " THE
N 485
490  END
560  REM  ...GRAFICO
565  DEF  FN N(X) = 1 / (4.4429
 * 2.718 ^ ((X * X) / 2))
570  FOR X = 1 TO 280 STEP 2
580  HPLOT X,158 - 660 *  FN N(
(X - 140) / 24)
590  NEXT X: RETURN
```

```
450 REM .... DISTR.NORMAL
460 DIM G(30)
470 PCLS:SCREEN 1,0:LINE(0,0)-(
0,191),PSET:LINE-(255,191),PSET
480 GOSUB 560
485 A$=INKEY$:IF A$<>" " THEN 4
85
560 DEFFND(X)=1/(4.4429*2.718^(
(X*X)/2))
570 COLOR 2,3:DRAW"BM2,190":FOR
 X=2 TO 255 STEP 2
580 LINE-(X,191-640*FND((X-127)
/24)),PSET
590 NEXT:RETURN
```

```
450 REM Distr.Normal
460 DIM g(30)
470 PLOT 4,150: DRAW 0,-140:
DRAW 245,0
480 GOSUB 560
485 IF INKEY$<>CHR$ 32 THEN
GOTO 485
560 REM Graf.
570 PLOT 4,10: FOR x=0 TO 1200
STEP 20
580 DRAW 4,600*FN n(ABS ((x-
600)/140))+10-PEEK 23678
590 NEXT x: RETURN
600 DEF FN n(x)=1/(PI*1.4142*
2.718^((x^2)/2))
```

Desta vez, rode o programa e selecione o teste 4, que é o de uma curva de distribuição do tipo normal. A linha 460 dimensiona uma matriz que será usada posteriormente na contagem das caras obtidas.

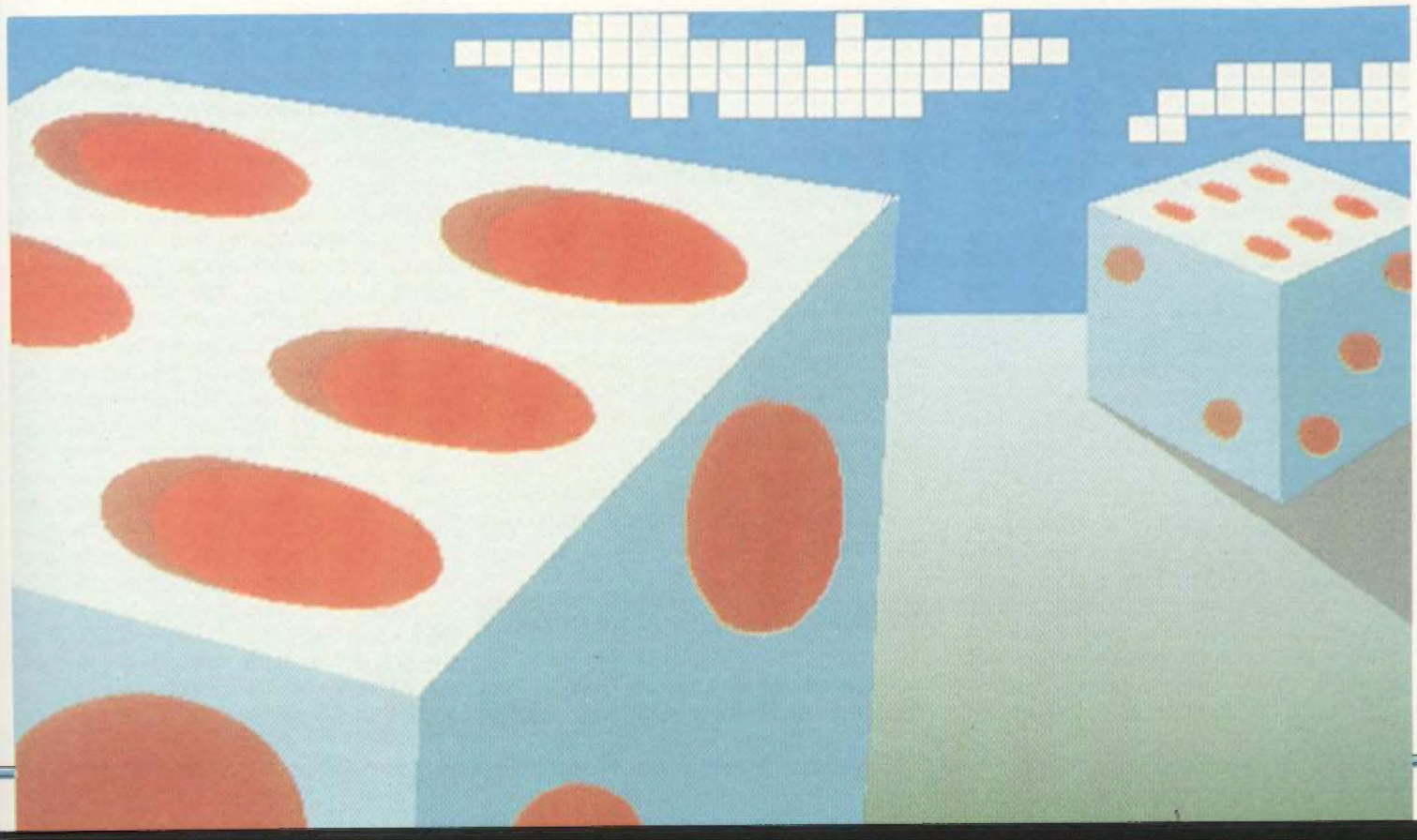
A linha 470 desenha os eixos das coordenadas X e Y e a linha 480 chama a rotina que desenha a curva. Essa rotina usa uma função matemática (linha 580) para desenhar a curva, o que explica a forma "perfeita" desta última. A função é definida pelas linhas 560 (MSX e TRS-Color), 565 (Apple e TK-2000) e 600 (Spectrum). Por enquanto, porém, não aperte nenhuma tecla, pois a rotina está incompleta. Aguarde o desenvolvimento do programa.

Quando usamos informações reais, torna-se difícil plotar uma curva contínua. Isso já era de se esperar, pois estamos lidando com probabilidades e não com certezas. A probabilidade de um resultado, tal como o de chover na Índia na época das monções, pode ser alta; mas, no caso do exemplo, têm sido registrados períodos em que a seca toma o lugar das chuvas.

O próximo teste ilustra muito bem esse ponto. Ele repete várias vezes os experimentos anteriores das moedas e coloca no gráfico (plota) os resultados obtidos. Digite então a segunda parte do teste 4:

```
490 FOR GM=1 TO 500
500 GOSUB 610
510 G(H)=G(H)+1
520 PSET(H*8+7,192-G(H)*2),1
530 NEXT
540 GOTO 230
```

```
490  FOR GM = 1 TO 200
500  GOSUB 610
510 G(H) = G(H) + 1
520  HPLOT 20 + 8 * H,160 - G(H
) * 4
530  NEXT GM
540  GOTO 230
```



```
490 FOR GM=1 TO 500
500 GOSUB 610
510 G(H)=G(H)+1
520 PSET(H*8+7,192-G(H)*2,3)
530 NEXT
540 GOTO 160
```

```
490 FOR g=1 TO 200: LET gm=g
500 GOSUB 610
510 LET g(h)=g(h)+1
520 PLOT 8+8*h,10+4*g(h)
530 PRINT AT 21,27;h;" "
540 NEXT g
550 STOP
```

Agora, rode novamente o quarto teste. Quando a curva ideal tiver sido desenhada, pressione a tecla de espaço para iniciar o lançamento. Aparecerá então uma série de pontos "crescendo" para preencher o espaço compreendido pela curva. Quando o teste estiver completado, quinhentos pontos (duzentos no Spectrum e no Apple) terão sido colocados no gráfico (especificados na linha 490). Em alguns micros, tal como no Spectrum, o tempo de execução desse teste é de alguns minutos. Para acelerá-lo, costuma-se inserir desvios (**GOTO**) em lugares apropriados ao longo do programa, mas na maioria das vezes isso traz alteração no funcionamento dos outros testes.

Esta seção do programa chama a rotina (linha 500) que lança a moeda trinta vezes (no Apple e no Spectrum, os lançamentos são mostrados na tela, como no terceiro teste). Cada bateria de trinta lançamentos forma um jogo, e pode resultar em certo número de caras entre 0 e 30. Portanto, na matriz da linha 460, **H** pode variar de 0 a 30. Se analisarmos pela óptica da lei das probabilidades, veremos que muito dificilmente **H** assumiria valor 0 ou valor 30 numa bateria de trinta lançamentos. Uma situação assim é conhecida como "ocorrência rara", justamente pela maneira com que os números aleatórios são gerados. Na verdade, números extremamente pequenos ou extremamente grandes de caras em lançamentos como esse não acontecem com freqüência. Sua ocorrência é *possível*, mas a probabilidade é muito baixa.

A linha 510 faz a contagem do resultado de cada jogo por intermédio de uma matriz. Por exemplo, toda vez que o resultado de um determinado jogo for onze caras, o elemento de matriz **G(11)** será incrementado de 1. Da mesma maneira, toda vez que o resultado for quinze caras, **G(15)** será também incrementado de 1. Inicialmente, todos os elementos da matriz são iguais a zero.

Após cada jogada, a linha 520 altera a escala do valor de **H** (número de caras em trinta lançamentos), a fim de compatibilizá-lo com os eixos X e Y. A próxima vez que ocorrer o mesmo resultado, um ponto será plotado na mesma coordenada X, mas uma unidade acima na coordenada Y.

A linha 530 faz a contagem do **H** em cada jogada, verificando também o número de jogadas.

COMO USAR A CURVA

Rode algumas vezes o teste 4 para ter uma idéia de como variam os pontos sob a curva; depois, rode-o com menores valores finais para **GM** na linha 490. Mesmo sem a curva ideal, seremos logo capazes de imaginar uma curva idealizada passando pelos picos. Na prática, o inverso desse processo imaginário é extremamente importante, pois quando conhecemos o perfil de uma curva podemos fazer a previsão do resultado de testes futuros.

O valor de **H** no pico central tem um interesse especial. Ele é a média dos 31 possíveis valores ao longo do eixo X (neste caso, 15). A média identifica o pico da curva, e constitui também o valor mais provável; sozinha, porém, ela não é muito informativa. Embora possamos dizer que 15 seja o resultado mais provável, 14 e 16 são quase tão prováveis quanto ele. Existe uma faixa de valores comuns em torno do pico, e é importante conhecer a largura desta faixa. A média é usada na especificação de um outro parâmetro estatístico importante — o desvio padrão —, que é a medida da largura da faixa de pontos comuns. A fórmula para o desvio padrão é relativamente complicada. Uma vez calculado esse parâmetro, podemos atribuir uma probabilidade para qualquer um dos pontos na curva.

O desvio padrão mede o quanto os valores variam nos dois lados da média. Por exemplo, uma seção de curva com desvio padrão de 1,96 em cada lado da média englobará 95% dos resultados. Se aumentarmos o desvio padrão para 2,58, a curva passará a conter 99% dos resultados.

Os pacotes comerciais de software estatístico calculam o desvio padrão com grande facilidade e rapidez.

SEIS RESULTADOS POSSÍVEIS

Existem muitos casos de eventos onde temos mais do que dois resultados possíveis. Em tais casos, calcular a chance de algum evento não é tão simples como olhar uma fileira do triângulo de Pascal.

Por exemplo, quando jogamos um dado, podemos conseguir um entre seis resultados. Se o dado não for viciado, todos os resultados são equiprováveis, ou seja, têm a mesma possibilidade de vir a acontecer.

Os resultados para o lançamento de dois dados podem ser calculados por intermédio da construção de uma tabela; mas, quanto mais resultados tivermos, mais complicado ficará o método que teremos de seguir.

Aqui está uma tabela com todos os resultados possíveis no lançamento de um par de dados:

		Valor do primeiro dado					
		1	2	3	4	5	6
Valor do segundo dado	1	2	3	4	5	6	7
	2	3	4	5	6	7	8
	3	4	5	6	7	8	9
	4	5	6	7	8	9	10
	5	6	7	8	9	10	11
	6	7	8	9	10	11	12

Como vemos na tabela, existem 36 maneiras diferentes de os dados caírem (seis linhas vezes seis colunas), embora haja somente onze resultados diferentes. A tabela apresenta várias outras informações. Existe uma possibilidade em 36 de obtermos os pontos mínimo e máximo (2 ou 12 aparecem somente uma vez na tabela); em contrapartida, há seis possibilidades em 36 (1/6) de obtermos 7 como resultado. Existe também uma chance em 36 de jogarmos repetido. Essa chance é mostrada na diagonal que vai de 2 (duas vezes o número 1) até 12 (dois 6).

Combinando o teorema binomial com essa tabela, podemos calcular as probabilidades para muitos jogos. Um bom exemplo de evento de múltiplo lançamento de dados é o jogo Monopólio, onde o jogador, caso esteja na "cadeia", tem direito a três lances para tentar conseguir uma dupla (dois dados iguais). Intuitivamente, pode parecer que ele conta com 50/50 de possibilidades de sair da "cadeia" (três jogadas, cada uma com 1/6 de chance), mas isso não é verdade. Pela tabela, podemos verificar que a chance de o jogador *não* conseguir uma dupla é de 30/36 (5/6) para cada jogada. Usando o teorema binomial, perceberemos facilmente que a chance de ele não conseguir uma dupla três vezes seguidas é 5/6 elevado à potência 3, ou 125/216. Isso equivale, aproximadamente, a 58% de possibilidades de fracasso.

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR LINHA

LINHA	FABRICANTE	MODELO
Apple II +	Appletronica	Thor 2010
Apple II +	CCE	MC-4000 Exato
Apple II +	CPA	Absolutus
Apple II +	CPA	Polaris
Apple II +	Digitus	DGT-AP
Apple II +	Dismac	D-8100
Apple II +	ENIAC	ENIAC II
Apple II +	Franklin	Franklin
Apple II +	Houston	Houston AP
Apple II +	Magnex	DM II
Apple II +	Maxitronica	MX-2001
Apple II +	Maxitronica	MX-48
Apple II +	Maxitronica	MX-64
Apple II +	Maxitronica	Maxitronic I
Apple II +	Microcraft	Craf II Plus
Apple II +	Milmar	Apple II Plus
Apple II +	Milmar	Apple Master
Apple II +	Milmar	Apple Senior
Apple II +	Omega	MC-400
Apple II +	Polymax	Maxxi
Apple II +	Polymax	Poly Plus
Apple II +	Spectrum	Microengenho I
Apple II +	Spectrum	Spectrum ed
Apple II +	Suporte	Venus II
Apple II +	Sycomig	SIC I
Apple II +	Unitron	AP II
Apple II +	Victor do Brasil	Elppa II Plus
Apple II +	Victor do Brasil	Elppa Jr.
Apple IIe	Microcraft	Craft IIe
Apple IIe	Microdigital	TK-3000 IIe
Apple IIe	Spectrum	Microengenho II
MSX	Gradiente	Expert GPC-1
MSX	Sharp	Hotbit HB-8000
Sinclair Spectrum	Microdigital	TK-90X
Sinclair Spectrum	Timex	Timex 2000
Sinclair ZX-81	Apply	Apply 300
Sinclair ZX-81	Engebras	AS-1000
Sinclair ZX-81	Filcres	NEZ-8000
Sinclair ZX-81	Microdigital	TK-82C
Sinclair ZX-81	Microdigital	TK-83
Sinclair ZX-81	Microdigital	TK-85
Sinclair ZX-81	Prologica	CP-200
Sinclair ZX-81	Ritas	Ringo R-470
Sinclair ZX-81	Timex	Timex 1000
Sinclair ZX-81	Timex	Timex 1500
TRS-80 Mod. I	Dismac	D-8000
TRS-80 Mod. I	Dismac	D-8001/2
TRS-80 Mod. I	LNW	LNW-80
TRS-80 Mod. I	Video Genie	Video Genie I
TRS-80 Mod.III	Digitus	DGT-100
TRS-80 Mod.III	Digitus	DGT-1000
TRS-80 Mod.III	Kemitron	Naja 800
TRS-80 Mod.III	Prologica	CP-300
TRS-80 Mod.III	Prologica	CP-500
TRS-80 Mod.III	Sysdata	Sysdata III
TRS-80 Mod.III	Sysdata	Sysdata Jr.
TRS-80 Mod.IV	Multix	MX-Compacto
TRS-80 Mod.IV	Sysdata	Sysdata IV
TRS-Color	Codimex	CS-6508
TRS-Color	Dynacom	MX-1600
TRS-Color	LZ	Color 64
TRS-Color	Microdigital	TKS-800
TRS-Color	Prologica	CP-400

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR NOME

FABRICANTE	MODELO	PAÍS	LINHA
Appletronica	Thor 2010	Brasil	Apple II +
Apply	Apply 300	Brasil	Sinclair ZX-81
CCE	MC-4000 Exato	Brasil	Apple II +
CPA	Absolutus	Brasil	Apple II +
CPA	Polaris	Brasil	Apple II +
Codimex	CS-6508	Brasil	TRS-Color
Digitus	DGT-100	Brasil	TRS-80 Mod.III
Digitus	DGT-1000	Brasil	TRS-80 Mod.III
Digitus	DGT-AP	Brasil	Apple II +
Dismac	D-8000	Brasil	TRS-80 Mod. I
Dismac	D-8001/2	Brasil	TRS-80 Mod. I
Dismac	D-8100	Brasil	Apple II +
Dynacom	MX-1600	Brasil	TRS-Color
ENIAC	ENIAC II	Brasil	Apple II +
Engebras	AS-1000	Brasil	Sinclair ZX-81
Filcres	NEZ-8000	Brasil	Sinclair ZX-81
Franklin	Franklin	USA	Apple II +
Gradiente	Expert GPC1	Brasil	MSX
Houston	Houston AP	Brasil	Apple II +
Kemitron	Naja 800	Brasil	TRS-80 Mod.III
LNW	LNW-80	USA	TRS-80 Mod. I
LZ	Color 64	Brasil	TRS-Color
Magnex	DM II	Brasil	Apple II +
Maxitronica	MX-2001	Brasil	Apple II +
Maxitronica	MX-48	Brasil	Apple II +
Maxitronica	MX-64	Brasil	Apple II +
Maxitronica	Maxitronic I	Brasil	Apple II +
Microcraft	Craft II Plus	Brasil	Apple II +
Microcraft	Caft IIe	Brasil	Apple IIe
Microdigital	TK-3000 IIe	Brasil	Apple IIe
Microdigital	TK-82C	Brasil	Sinclair ZX-81
Microdigital	TK-83	Brasil	Sinclair ZX-81
Microdigital	TK-85	Brasil	Sinclair ZX-81
Microdigital	TK-90X	Brasil	Sinclair Spectrum
Microdigital	TKS-800	Brasil	TRS-Color
Milmar	Apple II Plus	Brasil	Apple II +
Milmar	Apple Master	Brasil	Apple II +
Milmar	Apple Senior	Brasil	Apple II +
Multix	MX-Compacto	Brasil	TRS-80 Mod.IV
Omega	MC-400	Brasil	Apple II +
Polymax	Maxxi	Brasil	Apple II +
Polymax	Poly Plus	Brasil	Apple II +
Prologica	CP-200	Brasil	Sinclair ZX-81
Prologica	CP-300	Brasil	TRS-80 Mod.III
Prologica	CP-400	Brasil	TRS-Color
Prologica	CP-500	Brasil	TRS-80 Mod.III
Ritas	Ringo R-470	Brasil	Sinclair ZX-81
Sharp	Hotbit HB-8000	Brasil	MSX
Spectrum	Microengenho I	Brasil	Apple II +
Spectrum	Microengenho II	Brasil	Apple IIe
Spectrum	Spectrum ed	Brasil	Apple II +
Suporte	Venus II	Brasil	Apple II +
Sycomig	SIC I	Brasil	Apple II +
Sysdata	Sysdata III	Brasil	TRS-80 Mod.III
Sysdata	Sysdata IV	Brasil	TRS-80 Mod.IV
Sysdata	Sysdata Jr.	Brasil	TRS-80 Mod.III
Timex	Timex 1000	USA	Sinclair ZX-81
Timex	Timex 1500	USA	Sinclair ZX-81
Timex	Timex 2000	USA	Sinclair Spectrum
Unitron	AP II	Brasil	Apple II +
Victor do Brasil	Elppa II Plus	Brasil	Apple II +
Victor do Brasil	Elppa Jr.	Brasil	Apple II +
Video Genie	Video Genie I	USA	TRS-80 Mod. I

UM LOGOTIPO PARA CADA MODELO DE COMPUTADOR

INPUT foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.
Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:

Sinclair ZX-81

TRS-80

TK-2000

MSX

Spectrum

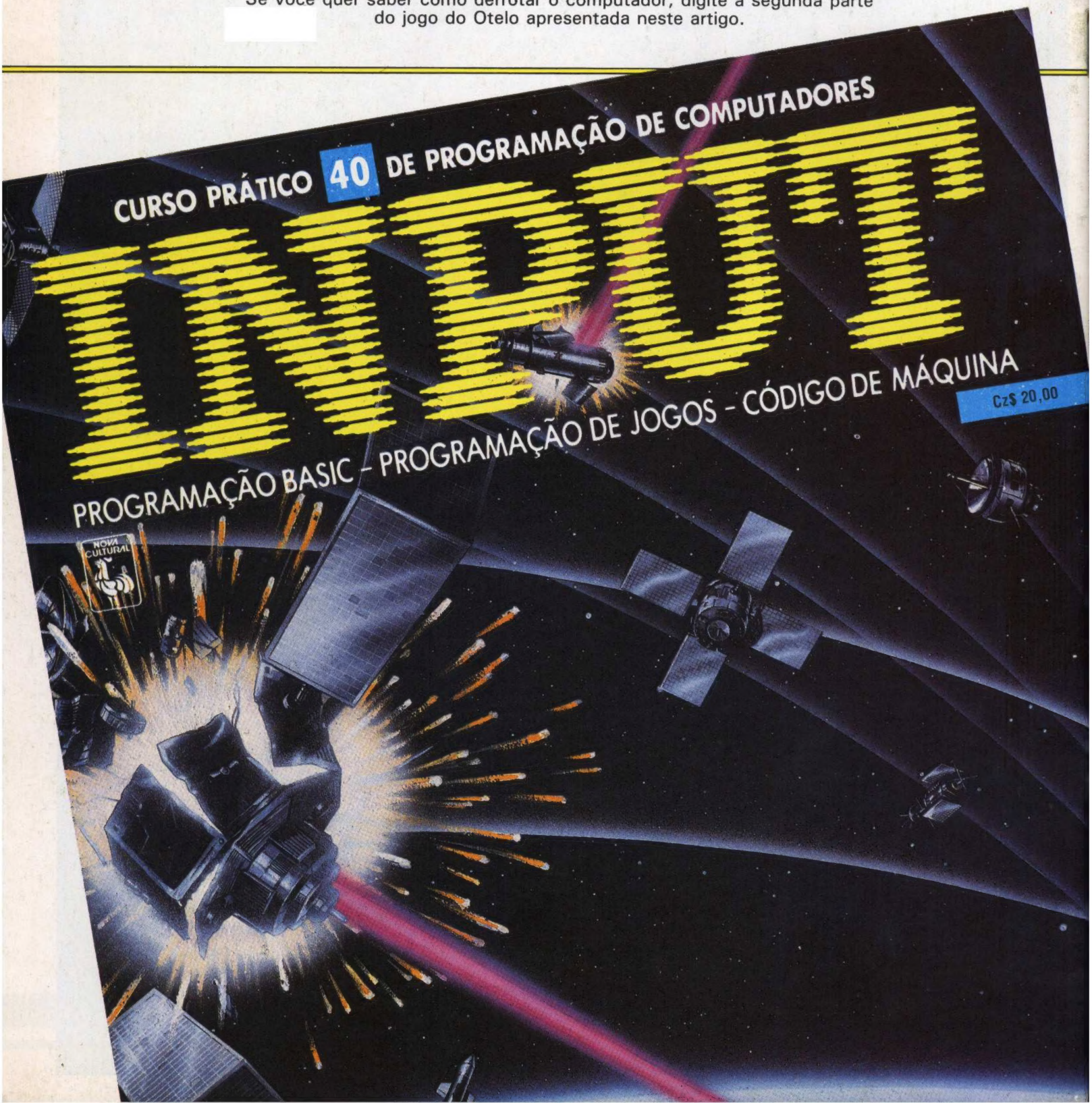
TRS-Color

Apple II

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

NO PRÓXIMO NÚMERO

PROGRAMAÇÃO BASIC
Atinja as estrelas, executando um programa que simula a trajetória de um corpo no espaço sideral.

CÓDIGO DE MÁQUINA
Pense em uma abertura musical digna do jogo *Avalanche*. Que tal um velho sucesso de 1560?

PROGRAMAÇÃO DE JOGOS
Se você quer saber como derrotar o computador, digite a segunda parte do jogo do Otelo apresentada neste artigo.